Л. А. Гладков, В. В. Курейчик,
В. М. Курейчик

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

УЧЕБНИК

Под редакцией В.М. Курейчика

ИЗДАНИЕ ВТОРОЕ, ИСПРАВЛЕННОЕ И ДОПОЛНЕННОЕ

*Допущено учебно-методическим объединением вузов
по университетскому политехническому образованию
в качестве учебника для студентов высших учебных заведений,
обучающихся по направлению
230100 «Информатика и вычислительная техника»,
специальности
230104 «Системы автоматизированного проектирования»*

МОСКВА
ФИЗМАТЛИТ®
2010

УДК 621.3+681.3
ББК 74.263.2
Г 52

Гладков Л. А., Курейчик В. В., Курейчик В. М. **Генетические алгоритмы** / Под ред. В. М. Курейчика. — 2-е изд., исправл. и доп. — М.: ФИЗМАТЛИТ, 2010. — 368 с. — ISBN 978-5-9221-0510-1.

Рассмотрены основные стратегии, принципы и концепции нового направления «Генетические алгоритмы». Описаны фундаментальные основы генетических алгоритмов и эволюционного моделирования. Проанализированы архитектуры генетического поиска и модели генетических операторов. Приведены конкретные примеры решения основных задач оптимизации на основе генетических алгоритмов и дано большое число контрольных вопросов и упражнений.

Для студентов вузов, обучающихся по направлению «Информатика и вычислительная техника», специальности «Информационные технологии в образовании», для специалистов, занятых разработкой интеллектуальных САПР, разработкой новых информационных технологий в науке, технике, образовании, бизнесе и экономике.

Допущено УМО вузов по университетскому политехническому образованию в качестве учебника для студентов вузов, обучающихся по направлению 210100 «Информатика и вычислительная техника», специальности 230104 «Системы автоматизированного проектирования».

ISBN 978-5-9221-0510-1

СОДЕРЖАНИЕ

Введение

> На базе фактов и законов (аксиом) с помощью логического вывода создаются научные теории. Однако особую роль играют аналогичные механизмы научного поиска, объединенные термином «интуиция».
>
> *А. Сухотин*

Большинство задач науки и техники относятся к обширному классу проблем поиска оптимальных решений, т. е. к оптимизационным задачам.

Часть задач оптимизации относится к классу комбинаторных и они, в большинстве случаев, имеют не одно, а множество решений различного качества. Существует множество алгоритмов для решения таких задач. Ядром всех комбинаторных алгоритмов являются операции полного или сокращенного перебора подмножеств решений. Стратегия комбинаторного перебора в различных алгоритмах реализуется по-разному. Для поиска лучшего решения, как правило, осуществляются направленный, случайный и комбинированный переборы всевозможных значений параметров задачи. В этой связи разрабатывается большое число точных, переборных и эвристических методов решения этих задач. До последнего времени не существовало эффективного механизма поиска решений на множестве альтернатив, что затрудняло получение качественных результатов за приемлемое время.

В конце 1960-х годов американский исследователь Джон Холланд в качестве принципов комбинаторного перебора вариантов решения оптимизационных задач предложил использовать методы и модели механизма развития органического мира на Земле. Поскольку основные законы эволюции живых организмов были исследованы генетикой, то и предложенный механизм получил название «генетические алгоритмы» (ГА). Первый ввел в обиход термин «генетический алгоритм» Д. Багли (США) в своей диссертации в 1967 г.

В мире сейчас успешно развиваются три основные научные школы по генетическим алгоритмам. К ним относятся американская, европейская и российская школы. В американской школе отметим таких

ученых, как Д. Холланд, Д. Гольдберг, Д. Коза, Л. Чамберс и др. В европейскую школу входят такие ученые, как Р. Клинг, П. Банерджи, Э. Фалькенауер и др. В российской отметим И. Букатову, Д. Батищева, И. Норенкова, авторов книги и многих др. Интерес к этой области исследований в мире с каждым годом возрастает. Для дальнейшего успешного развития этого направления в России необходимы учебники и монографии различных школ. Хочется надеяться, что этот пробел в ближайшее время будет восполнен.

Предлагаемое учебное пособие частично восполняет этот пробел. Учебное пособие предназначено для студентов, обучающихся по всем специальностям направлений «Информатика и вычислительная техника» и «Информационные системы», а также может быть полезно студентам других специальностей, желающим углубить свои знания в области методов оптимизации. Кроме того, пособие может быть полезно специалистам и инженерам соответствующих специальностей, оно может быть использовано при повышении квалификации и переподготовке специалистов по информационным технологиям.

Учебное пособие помимо оригинальных научных исследований авторов включает хорошо методически проработанные исследования Д. Холланда, Д. Голдберга, Дэвиса, Л. Чамберса (США), а также других российских и зарубежных ученых. Материалы каждого раздела пособия подготовлены на основе анализа существуюшей литературы, список которой приведен в конце каждого раздела. В учебном пособии содержатся материалы курсов лекций «Методы генетического поиска», «Эволюционное моделирование» и «Генетические алгоритмы», которые авторы читают в Таганрогском государственном радиотехническом университете. Частично материалы учебного пособия были апробированы при чтении лекций профессором В. М. Курейчиком в Мичиганском государственном университете (США) в 1992, 1993, 1994 годах.

Учебное пособие включает материал, который, в отличие от документаций на программные системы, содержит концепции, принципы, стратегии и методы генетического поиска, а также процедуры использования средств программных систем для решения конкретных задач проектирования на основе генетических алгоритмов.

Учебное пособие состоит из одной книги. Центральной частью пособия являются разделы 2–5, где приведены фундаментальные математические основы генетических алгоритмов, описаны процедуры генетического поиска, в которых шаг за шагом комментируется решение практических задач проектирования с момента постановки задачи до описания алгоритма и выполнения процедур работы.

Учебное пособие включает следующие разделы:

— генетика и основы эволюции (раздел 1);
— методы оптимизации (раздел 2);
— основные понятия и структура генетических алгоритмов (раздел 3);
— совместные схемы локального и генетического поиска (раздел 4);
— оптимизационные задачи на графах (раздел 5).

Комплексная цель и задачи изучения учебника

Цель учебника:

- дать представление о фундаментальных основах, общих принципах и законах, истории и основных этапах развития современной генетики, основных принципах теории эволюции Ч. Дарвина;
- изучить базовые формы и механизмы генетической изменчивости организмов, ознакомиться с основополагающими законами и принципами популяционной генетики и эволюционной изменчивости;
- рассмотреть основные математические модели процесса эволюции в естественных сообществах, изучить различные стратегии генетического поиска на базе различных моделей эволюции;
- дать представление об основных понятиях и определениях теории генетических алгоритмов, ознакомиться с различными моделями генетических алгоритмов;
- изучить структуру генетических алгоритмов, получить представление о простом генетическом алгоритме, рассмотреть основные гипотезы генетических алгоритмов,
- ознакомиться с основными видами генетических операторов, новыми модификациями основных генетических операторов;
- изучить базовые принципы и основные подходы к построению совместных схем локального и генетического поиска оптимальных решений;
- ознакомиться с наиболее распространенными архитектурами и стратегиями генетического поиска оптимальных решений;
- получить представление о генетическом программировании и его базовыми принципами, рассмотреть многоуровневые схемы организации генетического поиска.

В результате освоения учебника студент должен быть готов продемонстрировать следующие ***компетенции*** и ***уровень подготовки***:

1) знание основных естественнонаучных понятий, определений и терминов, заимствованных из биологии, генетики, теории эволюции и используемых в теории эволюционного моделирования, а также в процессе создания и практического использования генетических алгоритмов;
2) умение идентифицировать основные формы и механизмы генетической изменчивости, действующие в естественных организмах и системах;
3) навыки по составлению и модификации математических моделей, позволяющих симулировать эволюционные процессы, протекающие в естественных популяциях и биологических системах;
4) возможности использования методов формализации параметров и построения математической модели решаемой задачи;
5) знание основных определений и понятий теории генетических алгоритмов, базовой структуры генетических алгоритмов;

6) навыки по составлению математических моделей решаемых задач и подбору необходимых генетических операторов, выбору необходимой архитектуры и структуры генетического поиска;
7) возможности использования основных гипотез генетических алгоритмов для оценки вероятности «выживания» лучших решений в процессе генетического поиска;
8) знание основных методов и схем локального поиска;
9) умение в зависимости от специфики решаемой задачи строить новые или использовать существующие модификации основных генетических операторов;
10) навыки по модификации и составлению новых архитектур, стратегий и схем совместных локального и генетического поисков для решения практических задач оптимизации;
11) возможности использования различных моделей эволюции и уровней иерархии для повышения эффективности генетического поиска.

Самостоятельная работа с учебником предусматривает проработку лекций, подготовку к выполнению и сдаче лабораторных работ, тестирование, а также изучение литературы, формулировку цели работы, объекта и задач исследования, методов, источников и средств библиографического поиска, использованных для достижения поставленной цели.

Авторы благодарны членам научно-педагогической школы «Интеллектуальные САПР» Таганрогского государственного радиотехнического университета за обсуждение результатов работы. Выражаем искреннюю благодарность сотрудникам, аспирантам и студентам кафедры САПР за помощь в апробации алгоритмов и программ, ректору Таганрогского государственного радиотехнического университета Захаревичу В. Г. за поддержку исследований. Особо отметим работу рецензентов — профессоров Попова Э. В., Емельянова В. В., Еремеева А. П., чьи пожелания и замечания способствовали улучшению качества пособия.

Мы понимаем, что пособие не свободно от недостатков. Все замечания и предложения будут с благодарностью приняты.

Авторы

1. ГЕНЕТИКА И ОСНОВЫ ЭВОЛЮЦИИ

Все в биологии имеет смысл лишь в свете эволюционного учения.

Ф. Добжанский

Я понимаю под эволюцией все изменения (большие и малые), видимые и невидимые, адаптивные и неадаптивные.

М. Кимура

1.1. Краткие исторические сведения

1. Основоположником генетики является Иоганн Грегор Мендель, который в середине XIX в. открыл общий закон природы и вывел формулы расщепления признаков в гибридном потомстве. Известны три знаменитых закона И. Менделя:

1) ***закон однородности и реципрокности*** (взаимность, эквивалентность). Первое гибридное поколение оказывается полностью однородным;

2) ***закон расщепления***. При скрещивании гибридов первого поколения в потомстве происходит расщепление признаков по фенотипу 3:1, а по генотипу в отношении 1:2:1 [1]);

3) ***закон независимой комбинации***. Закон справедлив для потомков родителей, отличающихся более чем одной парой признаков, и говорит о том, что признаки наследуются независимо друг от друга.

И. Мендель пришел к выводу, что наследственность прерывиста (дискретна), что наследуется не большая совокупность свойств, а отдельные признаки. Он предположил, что в половых клетках есть какие-то материальные структуры (позже их назвали ***генами***), ответственные за формирование признаков. Одна из основных заслуг И. Менделя — его гипотеза о материальных задатках, которые в двойном комплекте находятся в клетках и которые зародыш получает от обоих родителей. Такие задатки (***хромосомы***) играют важную роль в явлениях наследственности. Они представляют собой нитевидные

[1]) Фенотип и генотип — см. глоссарий.

структуры, находящиеся в клеточном ядре. Каждый вид характеризуется вполне определенным числом хромосом, во всех случаях оно четное и их можно распределить попарно. При делении половых клеток и в процессе оплодотворения каждая хромосома ищет себе подобную хромосому. Они сближаются, располагаются параллельно друг другу и почти сливаются. Затем они расходятся. Но во время контакта почти все хромосомы обмениваются частями. Получается двуядерная клетка. После этого ядра сливаются и образуется одно ядро с нормальным двойным набором хромосом. Получившаяся клетка (***зигота***), многократно делясь, образует зародыш, из которого и развивается организм. Соответственно ***гетерозиготой*** называется особь, в которой все аллели отличаются, а ***гомозиготой*** — особь, в которой все аллели данного гена одинаковы.

2. В клетках существует сложный и важный механизм перераспределения хромосом. Каждая клетка организма имеет одинаковое число хромосом. У потомков содержится то же самое число хромосом, причем ровно половина от отца, а половина — от матери. Хромосомы передают наследственные признаки. Ч. Вильсон в 1900 г. определил, что хромосомы состоят из генов. Модель хромосомы в настоящее время — это нить, на которую, словно бусины, нанизаны гены. Определяют гомологичные и негомологичные хромосомы.

Гомологичными хромосомами называются пары одинаковых хромосом, несущие гены одинаковых признаков. Две гомологичные хромосомы (одна от отца, а другая от матери) сближаются при созревании половых клеток и обмениваются частями. Это явление назвали ***crossover*** (***кроссинговер***). Он происходит случайным образом между разными генами с разной частотой.

3. В 1809 г. Ж. Б. Ламарк определил ***эволюцию*** как возникновение новых форм путем постепенного изменения старых. При этом происходит усложнение форм. Он считал, что организмы изменяются под прямым воздействием окружающей среды, что причиной эволюции являются «упражнения» органов и внутреннее стремление к прогрессу.

4. Ч. Дарвин определил в живой природе существование общего принципа — ***естественного отбора***. Он различал две стороны эволюционного учения: учение о материале для эволюции и факторах. Движущая сила эволюции — естественный отбор.

5. Одним из основателей генетики является датский генетик В. Иогансен. Он впервые ввел термины ген, генотип, фенотип, аллель. ***Ген*** — независимая, комбинирующаяся и расщепляющаяся при скрещиваниях единица наследственности, отвечающая за появление какого-либо признака. ***Генотип*** — совокупность всех генов, находящихся в хромосомах организма. Он содержит всю наследственную информацию и передает ее от поколения к поколению на основе генетической программы. На ее основе формируется

совокупность всех внешних и внутренних признаков, характеризующая организм, называемая ***фенотипом***.

6. В настоящее время считается, что гены определяют механизм наследования. Он связан с отбором. Через отбор происходит направленное влияние окружающей среды на наследственную изменчивость. Отметим, что наследственная изменчивость случайна. Под воздействием окружающей среды отбираются признаки, которые лучше других соответствуют условиям жизни. Г. де Фриз наследственные изменения отдельных генов назвал ***мутациями***. Мутации служат элементарным материалом для эволюционного процесса.

В процессе эволюции в результате естественного отбора наследственных мутаций возникают адаптации. ***Адаптация*** — это процесс приспособления строения и функции организмов к условиям внешней среды. Считается, что развитие природы идет по пути возникновения все новых и новых механизмов адаптации.

Приведем факторы, которые меняют генетический состав природной ***популяции***:

1. ***Мутационный процесс***. Согласно закону Г. Харди и В. Вайнберга равновесие устанавливается после первого скрещивания. В каждом поколении возникают новые мутации, происходит скрещивание, вызывающее новое равновесие и т. д. Следовательно, благодаря мутационному процессу состояние равновесия непрерывно меняется.
2. ***Изоляция***. Чем меньше размер популяции, тем более вероятно, что в ней проявятся скрытые изменения.
3. ***«Волны жизни»***. Изоляция ведет к уменьшению величины популяции. Объем популяции может меняться в зависимости от окружающей среды, т. е. на основе обратной связи.
4. ***Отбор***. Факторы 1, 2, 3 — ненаправленные, они изменяют состав популяции случайным образом. Фактор 4 в отличие от них действует в определенном направлении, которое наилучшим образом соответствует условиям жизни.

7. Классическая генетика к началу 1940-х годов пришла к пониманию ***дискретности*** таких качеств, как наследственность и изменчивость. Это стало возможным в первую очередь благодаря формированию теории гена в работах школы Т. Моргана. Основные положения этой теории можно сформулировать следующим образом.

1. Все признаки организмов находятся под контролем генов.
2. Гены — элементарные единицы наследственной информации — находятся в хромосомах.
3. Гены могут изменяться, т. е. мутировать.
4. Мутации отдельных генов приводят к изменению отдельных элементарных признаков, или фенов.

8. В 1953 г. с работ М. Уоткена и Ф. Крика началась новая наука — молекулярная генетика. Биохимические особенности живых организмов наследуются по законам, которые открыл И. Мендель. Все

биохимические реакции в клетках управляются сложными белковыми веществами — ферментами. Существует следующая гипотеза: один ген — один фермент. Следовательно, функция гена состоит в образовании какого-либо одного фермента. В генах «записаны» планы строения белков — планы всех наследственных признаков. В процессах наследования признаков определяющую роль играет поведение хромосом при делении клеток.

9. В настоящее время ген определяют как структурную ***единицу наследственной информации***, далее неделимую в функциональном отношении. Его рассматривают как участок молекулы ДНК, кодирующий синтез одной макромолекулы или выполняющий какую-либо другую элементарную функцию. Комплекс генов, содержащихся в наборе хромосом одного организма, образует ***геном***. Ген — сложная структура, состоящая из мутирующих элементов, разделяемых при рекомбинации.

1.2. Кроссинговер

1. Рекомбинация сцепленных генетических факторов свойственна всем группам организмов, исследованным к настоящему времени. Генетическая рекомбинация реализует несколько типов перераспределения наследственных факторов:

- рекомбинация хромосомных и нехромосомных генов;
- рекомбинация целых негомологичных хромосом;
- рекомбинация участков хромосом, представленных непрерывными молекулами ДНК.

При решении задач оптимизации возможно моделирование процессов рекомбинации. В этом случае любое решение рассматриваемой задачи представляется как некоторая информация, способная к обновлению посредством введения элементов другого решения. В задачах оптимизации условно считают, что хромосомы являются закодированным представлением альтернативных решений. Хромосомы, представляющие собой отображения решений, должны быть гомологичны, так как являются взаимозаменяемыми альтернативами. Новый механизм решения оптимизационных задач в отличие от существующих механизмов осуществляет не замену одного сгенерированного решения на другое, что осуществимо простой оценкой исходных решений в соответствии с принятым критерием, а получение новых решений посредством обмена между ними информацией.

Каждый участок хромосомы (альтернативного решения) несет определенную функциональную нагрузку. Желательно создание такой комбинации участков хромосом, которая составляла бы наилучшее из решений, возможное при исходном генетическом материале. Поэтому целью рекомбинации является накопление в конечном решении всех лучших функциональных признаков, какие имелись в наборе исходных решений.

2. Рассмотрим рекомбинацию участков хромосом, представленных непрерывными молекулами ДНК. Здесь может быть выделено несколько подтипов рекомбинации:

- *регулярная* (общая) *рекомбинация*, представляющая собой кроссинговер, т. е. обмен гомологичными участками в различных точках гомологичных хромосом, приводящий к появлению нового сочетания сцепленных генов;
- *спрайт* — специфическая рекомбинация, т. е. обмен генных носителей, часто разных по объему и составу, на коротких специализированных участках;
- *нереальная рекомбинация*, реализующая негомологичные обмены. Негомологичный обмен в целом не представляет интереса, так как появляются нереальные решения исследуемой задачи.

Схему реализации кроссинговера покажем на рис. 1.1.

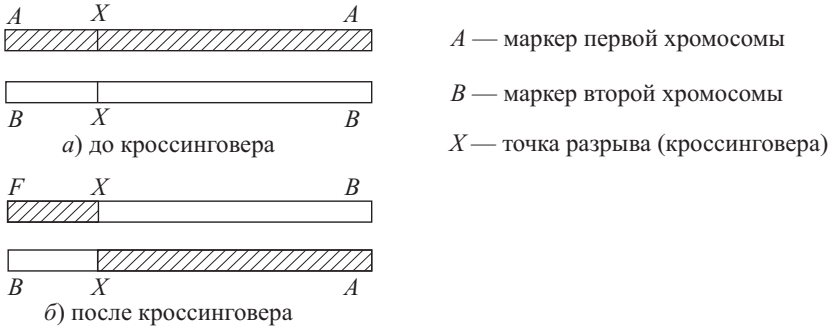


Рис. 1.1. Схема реализации кроссинговера

Т. Морган предположил, что кроссинговер может происходить не только в одной, но и в двух и даже большем числе точек. На рис. 1.2 представлена экспериментально установленная схема двойного кроссинговера между хромосомами (w и f).

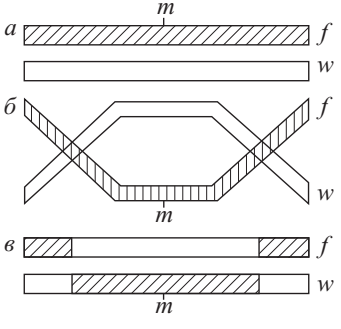


Рис. 1.2. Схема двойного кроссинговера: *а*) до кроссинговера; *б*) во время кроссинговера; *в*) после кроссинговера

Как видно на рис. 1.2, *б*, точки разрыва определяются в местах пересечения хромосом. Первая хромосома, потомок, содержит по краям две части хромосомы f и в центре — одну часть из хромосомы w. Вторая хромосома, потомок, содержит по краям две части хромосомы w и в центре — одну часть из хромосомы f. В настоящее время существует большое число схем реализации механизма кроссинговера. Таким образом, в общей схеме реализации кроссинговера

можно выделить два варианта обмена информацией. В первом варианте используется одна точка разрыва, а во втором — две и более точек разрыва.

1.3. Мутация

1. По Г. Менделю, расщепление при независимом наследовании и при кроссинговере определяет генетическую изменчивость организмов вследствие комбинаторики существующих генов (аллелей). ***Генетическая изменчивость*** — это изменения, произошедшие в структуре генотипа и передаваемые по наследству. ***Изменчивость*** — разнообразие признаков и свойств у особей и групп особей любой степени родства. Различают изменчивость направленную и ненаправленную.

2. ***Направленная***, или ***определенная***, ***изменчивость*** обычно массовая и приспособительная. В данном случае наследуется не изменение признака, а способность к изменению, но в разной степени. Определенная изменчивость есть продукт эволюции, способность к ней возникает в результате отбора в течение многих поколений. Но само изменение признака под влиянием какого-нибудь фактора внешней среды исчезает с гибелью организма, потомки должны обретать его заново. Только в этом смысле определенная изменчивость является ненаследуемой. Это — наследственность, проявляющаяся в фенотипе не всегда, а лишь при воздействии определенного фактора внешней среды.

3. ***Ненаправленная***, или ***неопределенная***, ***изменчивость*** возникает независимо от природы вызвавшего ее фактора, причем изменяющийся признак может изменяться и в сторону усиления, и в сторону ослабления. При этом она не массовая, а единичная. Различают два типа неопределенной изменчивости — *комбинативную* и *мутационную* изменчивость.

Источником комбинативной изменчивости являются новые сочетания материнских и отцовских хромосом, возникающие при образовании потомства. При этом хромосомы иногда обмениваются частями (кроссинговер), так что число комбинаций генов в каждом новом поколении резко возрастает. Возможное число генотипов определяется выражением

$$g = \left[\frac{r(r+1)}{2}\right]^m,$$

где r — число аллелей, m — число генов.

Мутационная изменчивость — процесс изменения генетической структуры организма, его генотипа. При этом изменяется число хромосом, или их строение, или же структура слагающих хромосому генов. Как и комбинативная изменчивость, мутационная — процесс ненаправленный (признаки могут изменяться случайным образом), немассовый (одновременное возникновение какой-нибудь одной мутации у целого

ряда особей в популяции невозможно) и неприспособленный повышать и понижать жизнеспособность их носителей.

Неопределенная изменчивость — материал для процесса эволюции. Изменения организмов, по Ч. Дарвину, отбираются факторами внешней среды. При этом с большей вероятностью выживают и оставляют потомство носители полезных в данной среде признаков, возникших в результате мутации или рекомбинации определяющих эти признаки генов.

Мутационная изменчивость связана с процессом образования мутаций. ***Мутация*** — генетическое изменение, приводящее к качественно новому проявлению основных свойств генетического материала. Считается, что при мутации происходят внезапные скачкообразные изменения в структуре генотипа.

4. Мутационная теория создана Г. де Фризом. Ее основные положения следующие:

- мутации — дискретные изменения наследственности;
- мутации в природе спонтанны;
- они передаются по наследству;
- мутации встречаются редко;
- они могут быть вредными, полезными, нейтральными и т. п.

5. Существует много систем классификации мутаций, их можно разделить

1) по способу возникновения: спонтанные мутации, индуцированные мутации. Возникают редко. Процесс зависит от внутренних и внешних факторов;
2) по адаптивному значению выделяют мутации: положительные, отрицательные, нейтральные;
3) по изменению генотипа мутации бывают: генные (точковые), хромосомные, геномные;
4) по локализации в клетке: ядерные, цитоплазматические.

6. В задачах оптимизации, как считают авторы, наибольший интерес представляют модели мутаций третьего вида. В этой связи рассмотрим их подробнее.

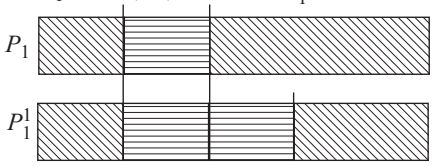
6.1. ***Генные мутации*** затрагивают, как правило, один или несколько генов. При этом один ген (или их последовательность) может превратиться в другой, может выпасть либо дублироваться, а группа генов может развернуться на 180 градусов.

6.2. ***Хромосомные мутации*** приводят к изменению числа, размеров и организации хромосом. Существуют внутрихромосомные перестройки и межхромосомные перестройки.

Внутрихромосомные перестройки подразделяются на дупликации, делеции и инверсии:

• ***дупликация***, т. е. удвоение. При этом один из участков хромосомы представляется более одного раза. Например:

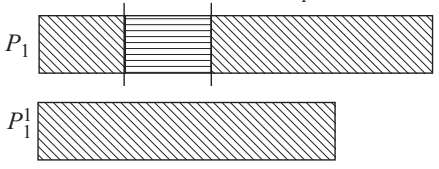
$P_1: A\,B\,C\,|\,D\,|\,E\,F\,G\,H \rightarrow P_1^1 : A\,B\,C\,D\,D\,E\,F\,G\,H$



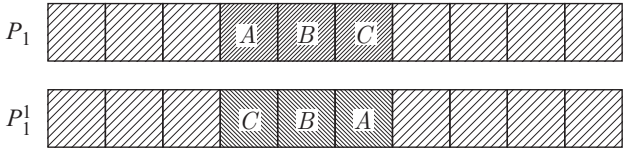
здесь и далее метка | — означает точку или место разрыва хромосомы.

• ***делеция***, т. е. удаление внутреннего участка хромосомы. Например:

$P_1: A\,B\,C\,|\,D\,E\,|\,F\,G\,H \rightarrow P_1^1 : A\,B\,C\,F\,G\,H$

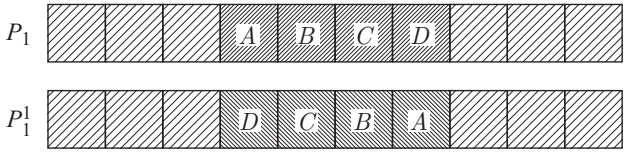


• ***инверсии*** — повороты участка или всей хромосомы на 180 градусов. Инвертированный участок при нечетной длине хромосомы включает центральный ген (перецентрическая инверсия) или не включает его при четной длине хромосомы (парацентрическая). Например, перецентрическая инверсия запишется так:



здесь длина участка инвертированной хромосомы $L(P_1) = 3$.

А парацентрическая инверсия, например, такова:



здесь $L(P_2) = 4$.

Межхромосомные перестройки часто называют ***транслокациями***. При этом участок хромосомы перемещается (транслоцируется) на другое место хромосомы. Выделяют следующие типы транслокаций: реципрокные, нереципрокные (транспозиции), децентрические (полицентрические) и центрические (робертсоновские).

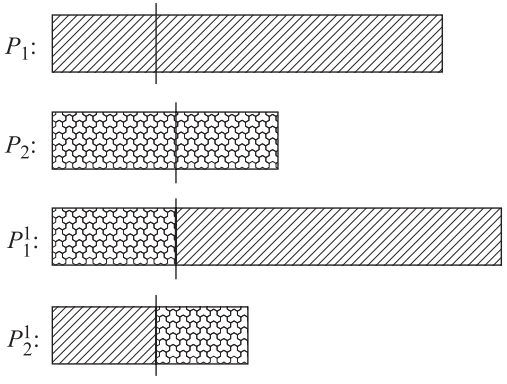
Реципрокные — взаимный обмен участками негомологичных хромосом. В отличие от кроссинговера, при транслокации происходит обмен участков хромосом различной длины. Например,

P_1: $AB|CDEFGH$

P_2: $MNO|PQR$

P_1^1 : $MNOCDEFGH$

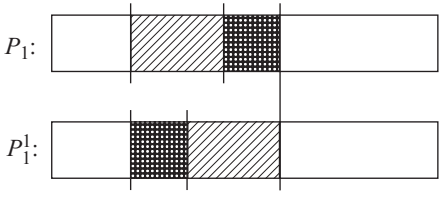
P_2^1 : $ABPQR$



Нереципрокные (***транспозиции***) — участок хромосомы изменяет свое положение или включается в другую хромосому без взаимного обмена. Например,

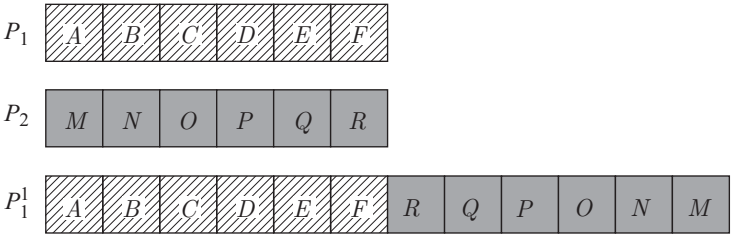
P_1: $A|BC|DEF|GH$

P_1^1 : $ADEFBCGH$



Дицентрические (***полицентрические***) — слияние двух (или более) фрагментов хромосом, несущих участки с центромерами.

Центрические (***робертсоновские***) — происходят при слиянии двух центромер хромосом с образованием одной мета- или субметацентрической хромосомы при инверсии второй хромосомы. Например:



7. В естественных системах роль мутаций заключается в том, что именно они генерируют новые функции, затем происходит дупликация, закрепляющая обе функции, а далее начинается отдельная эволюция исходной и новой функций. Мутация при постоянстве целевой функции оказывается решающим фактором адаптации. Установлено, что в неизменных условиях биологические виды дают более высокую адаптивность.

В общем же эволюция стремится к усреднению (так как происходит все более однородное смешивание разного по качеству генетического материала). Поэтому для получения наилучших результатов в качестве одного из методов используется селекция.

1.4. Селекция

1. Применение селективных методов означает методический отбор генетического материала в соответствии с принятым критерием, что должно повысить скорость получения конечного результата и его качество. Отметим, что ускорение мутационного процесса ведет к получению все более разнообразного генетического материала.

Селекция представляет собой форму искусственного отбора, где в отличие от естественного отбора эволюция направляется факторами внешней среды. Селекция как наука создана Ч. Дарвином, который выделял три формы отбора:

1) естественный отбор, вызывающий изменения, связанные с приспособлением к новым условиям;
2) бессознательный отбор, при котором в процессе эволюции сохраняют лучшие экземпляры;
3) методический отбор, при котором проводится целенаправленное изменение популяций в сторону установленного идеала.

В учении об отборе Ч. Дарвин показал, что основой является отбор наилучших форм, требующий таких условий, как:

- полный выбор исходного материала;
- правильная постановка цели;
- проведение селекции в достаточно широких масштабах и возможно более жесткий отбор материала;
- отбор по одному основному признаку.

2. ***Естественный отбор*** — процесс, направленный к повышению (или понижению) вероятности оставления потомства одной формой организмов по сравнению с другими. Отбор, прежде всего, действует в пределах каждой популяции, отбирая те или иные входящие в ее состав генотипы.

При относительной стабильности внешних условий преобладающие генотипы все время будут сохранять свое доминирующее положение. Напротив, все уклонения от этой группы будут уничтожаться. Такая форма отбора названа *центростремительным* или *стабилизирующим*

отбором. Однако при изменении условий существования может произойти отбор, ведущий к замене одних количественно преобладающих генотипов другими. Эта *форма отбора* названа *движущей*, или *ведущей*.

Естественный отбор — единственный направляющий эволюцию элементарный фактор. Его действие всегда направляется складывающимися условиями существования.

Рассмотренные выше факторы по степени влияния на эволюцию можно упорядочить следующим образом:

1) естественный отбор;
2) изоляция;
3) колебание численности популяции;
4) мутационные процессы.

3. В современной селекции широко применяют массовый и индивидуальный отбор. Для создания генетически устойчивых видов, обладающих желаемыми признаками, необходимо знать генетическую структуру исходных популяций. Массовый отбор, производимый по фенотипу, является менее эффективным, чем индивидуальный.

При индивидуальном отборе популяцию делят на отдельные части, причем для размножения выбирают носителей желаемых свойств. Иногда отбор ведут по боковым родственникам.

Инбридинг — близкородственное скрещивание — позволяет быстро выделить части, несущие желаемые гены. Здесь применяется жесткий отбор экземпляров, а также введение в части популяции нового генетического материала.

Рассмотрим схему селекции. Вначале создают несколько частей популяций, несущих желаемые признаки. Затем из них получают межлинейные гибриды первого поколения. Экземпляры, дающие наилучшие комбинации признаков, отбирают для последующего размножения. При скрещивании различных видов гибриды первого поколения часто превосходят лучшую из родительских форм. В последующих поколениях, даже уже во втором, эти признаки ослабляются и потом затухают. Их закрепление в поколениях — одна из важнейших генетических задач селекции.

4. Подобную схему селекции можно использовать при решении оптимизационных задач. В соответствии с принятым критерием даем оценку каждому из альтернативных решений, отбираем и закрепляем те решения, которые получили наиболее высокую оценку.

Генетический материал обладает такими универсальными свойствами, как дискретность, непрерывность, линейность и относительная стабильность, выявляемыми в ходе генетического анализа.

Дискретность — существование гена, хромосомы, генома — проявляется в виде множества аллелей, соответствующих гену, множества генов, соответствующих группе сцепления (хромосоме), множества групп сцепления, соответствующих геному. Это значит, что можно

выделять в исходном генетическом материале отдельные фрагменты, контролирующие те или иные функции.

Непрерывность — физическое единство групп сцепления — проявляется в существовании множества генов, обнаруживающих сцепление между собой. Это свойство находит выражение также в различных типах эффекта положения гена. Это свойство означает, что определенные комбинации генов совместно контролируют некоторую функцию.

Линейность — одномерность записи генетической информации — проявляется в определенной последовательности генов в пределах группы сцепления.

Относительная стабильность генетического материала — его способность к изменениям — проявляется в мутациях, что означает способность генов к воспроизведению и изменению с последующим воспроизведением измененных вариантов.

Повышение разрешающей способности генетического анализа возможно с помощью:

- анализа большого числа особей;
- применения кроссинговера;
- ускорения мутационного процесса;
- применения селективных методов.

Все эти приемы имеют большое значение в живой природе и могут найти применение при решении технических задач. Легко видеть, что увеличение числа хромосом (решений) приводит к росту разнообразия генетического материала, что означает увеличение исходного набора контролируемых генами функций, а это, в свою очередь, позволяет провести более широкий отбор альтеранатив и тем самым улучшить конечное решение.

1.5. Особенности механизма эволюционной адаптации

1. Основой эволюционных процессов в органическом мире служит популяция. ***Популяция*** — это многочисленная совокупность особей определенного вида, в течение длительного времени (большого числа поколений) населяющих определенный участок географического пространства, внутри которого осуществляется та или иная степень случайного свободного скрещивания. Надстройкой популяции является вид.

Вид состоит из нескольких популяций, каждая из которых эволюционирует самостоятельно. В популяции нет тождественных особей. Каждая особь является носителем уникального генотипа, который управляет формированием фенотипа.

Элементарным носителем генотипа в организме каждой особи является клетка. Но поскольку жизнь клетки намного меньше жизни особи, то фактическим носителем генотипа является череда поколений клеток. Рождение новых поколений клеток обеспечивается биосинтезом клеток. В процессе биосинтеза генотип особи не меняется.

2. Существование каждой особи ограничено некоторым интервалом времени, по завершении которого особь погибает. При этом генотип особи исключается из генофонда популяции. Существенным фактором является то, что при жизни особь может передать наследственную информацию — генотип — потомству. Существуют два ***механизма передачи наследственной информации*** при рождении потомства:

- ***бесполое размножение***; основано на равном делении ядер клеток, при котором происходит простое копирование наследственной информации без какого-либо изменения;
- ***половое размножение***; основано на редукционном и последующем равном делениях ядер клеток двух родителей, в результате чего потомок получает копии наследственной информации обоих родителей.

В настоящее время установлено, что наследственные факторы — гены — являются специфическими молекулами ДНК или функционально обособленными участками таких молекул и что устойчивая передача генов от родителей к потомкам зависит, в первую очередь, от способности молекул ДНК к авторепродукции.

3. Участки молекул, занимаемые отдельными генами, называются ***локусами***. Для многих локусов известно не одно, а несколько устойчивых состояний генов. Такие состояния генов называют аллеломорфными генами, аллеломорфами, или просто ***аллелями***. Например, моделью, поясняющей понятия локусов и аллелей, может быть следующая запись:

A	B	C	D	E	F
1	2	3	4	5	6

Здесь A, B, C, D, E, F — код гена; ген A расположен в локусе 1; ген B — в локусе 2; C — 3 и т. д., причем гены $ABCDEF$ могут принимать устойчивые различные состояния, например, $A = \{00, 01, 11\}$.

4. Каждая особь в течение жизни подвергается воздействиям внешней среды. В некоторых случаях эти воздействия могут привести к перестройкам молекул ДНК, переносящих наследственную информацию. Изменение первоначальной последовательности генов в молекулах ДНК приводит к изменению свойств этой молекулы, а, следовательно, и наследственной информации. Изменение наследственной информации в течение жизни одной особи, как мы уже отмечали, является мутацией, а сама особь — мутантом. Частота возникновения отдельных мутаций всегда относительно низка — 10^{-4}–10^{-6} на поколение. Мутационный процесс осуществляет неопределенную изменчивость.

Внешние воздействия вызывают мутации отдельных индивидов. Закрепление генотипа мутанта в популяции происходит при скрещивании

этого мутанта с другой особью и образовании потомства. В малочисленной популяции в результате свободного скрещивания генотип мутанта быстро размножается и вызывает резкие изменения в структуре генофонда популяции. И, наоборот, в достаточно многочисленной популяции генотип мутанта вскоре «растворяется» среди других генотипов.

Таким образом, мутационный процесс приводит к изменению (реорганизации) наследственной информации, а ***колебание численности*** управляет «давлением» возникающих мутаций на генофонд популяции. Эти два фактора являются своего рода «поставщиками» материала для эволюции.

Рассмотрим известную модель прогноза численности популяции, приведенную Ф. Хедриком. Пусть P^t и P^{t+1} — численности популяций в поколениях t и $t+1$. Тогда отношение численностей популяциий в поколениях $t+1$ и t запишется так:

$$R = \frac{N_{t+1}}{N_t}. \tag{1.1}$$

Величину R иногда называют скоростью замещения или средним числом потомства для одного родителя. Если $R = 2$, то в каждом следующем поколении численность популяции удваивается. Если $R = 3$, то утраивается.

Из выражения (1.1) получим

$$N_{t+1} = RN_t. \tag{1.2}$$

Следовательно, численность последующего поколения зависит от численности предыдущего поколения и скорости замещения. Если предположить, что (1.2) справедливо и для следующих поколений, то

$$N_{t+2} = RN_{t+1}$$

или

$$N_{t+2} = RRN_t = R^2N_t.$$

Итак, начальная численность N_0 популяции P^0 связана с численностью популяции в поколении t следующим образом:

$$N_t = R^tN_t.$$

Например, пусть первоначально численность популяции равна 100 особям ($N_0 = 100$), а скорость замещения равна 3, тогда предполагаемая численность популяции через 5 поколений составит

$$N_5 = 3^5 \cdot 100 = 243\,000.$$

При тех же условиях, но при $R = 2$ получим

$$N_5 = 2^5 \cdot 100 = 32\,000.$$

Можно определить разницу изменения численности популяции в разные отрезки времени:

$$\Delta N = RN_t - N_t = N_t(R-1).$$

Если происходит непрерывный рост численности популяции, то изменение ее численности описывают дифференциальным уравнением

$$\frac{\partial N}{\partial T} = rN,$$

здесь r — мгновенная скорость роста, величина, характеризующая рост численности популяции.

5. Под эволюцией понимаются медленные, постепенные количественные и качественные изменения объекта. При этом каждое новое состояние объекта, как правило, имеет по сравнению с предыдущим более высокий уровень развития и организации. Эволюция приводит к формированию адаптаций (приспособлений) организмов к условиям их существования. В науке под адаптацией понимают процесс накопления и использования информации в системе, направленный на достижение ее (системы) оптимального состояния, при первоначальной неопределенности и изменяющихся внешних условиях.

В связи с развитием теории эволюции данные идеи все более используются при моделировании мышления и поведения человека, создании систем принятия решений в неопределенных и нечетких условиях, решении оптимизационных задач большой размерности. Для описания концепций популяционной генетики Ф. Хедрик предлагает использовать различные модели. Они представляют собой вербальное, графическое или математическое описание событий. Преимущество моделей перед простым описанием состоит в том, что они охватывают как сам процесс, так и его составные части. Из многочисленных теорий эволюционного развития авторы планируют использовать и рассматривать шесть моделей (видов) эволюции.

5.1. Модель эволюции Дарвина — это условная структура, реализующая процесс, посредством которого особи некоторой популяции, имеющие более высокое функциональное значение, получают большую возможность для воспроизведения потомков, чем «слабые» особи. Такой механизм часто называют ***методом «выживания сильнейших»***. ***Движущими силами эволюции по Ч. Дарвину*** являются:

- ***неопределенная изменчивость***, т. е. наследственно обусловленное разнообразие организмов каждой популяции;
- ***борьба за существование***, в ходе которой устраняются от размножения менее приспособленные организмы;
- ***естественный отбор*** — выживание более приспособленных особей, в результате которого накапливаются и суммируются полезные наследственные изменения и возникают новые адаптации.

Существует спектр путей развития, по которым может пойти эволюция Дарвина. Возможный путь развития здесь определяет случайность. Эволюция по Дарвину состоит из следующих положений:

- в природе все подвержено неопределенной наследственной изменчивости, производится потомство, отличающееся по многим признакам;
- все организмы в природе размножаются в геометрической прогрессии, но численность всех организмов в среднем остается более или менее постоянной, она колеблется около средней величины;
- основой отбора является метод «выживания сильнейших».

Моделирование эволюции может предоставить алгоритмические средства для решения оптимизационных задач. В общих чертах, эволюция может быть описана как многоступенчатый итерационный процесс, состоящий из случайных изменений и последующей затем селекции. Таким образом, существует взаимосвязь между определением эволюции и оптимизационными алгоритмами. На рис. 1.3 приведена условная ***упрощенная схема модели эволюции Дарвина***.

Рис. 1.3. Условная упрощенная схема модели эволюции Дарвина

На основе данной схемы можно составить примерный алгоритм модели эволюции по Дарвину.

1. *Популяция*. Пусть существует популяция особей (например, табун лошадей) на некоторой территории обитания (степь). Табун (популяция) насчитывает 50 взрослых лошадей (особей).
2. *Наследственность*. Каждый год в табуне (популяции) рождаются жеребята (новое поколение, потомство).
3. *Изменчивость*. Все родившиеся жеребята (потомки) отличаются друг от друга и от родителей цветом, размерами, ростом, физическими данными (скорость, выносливость и т. д.).
4. *Отбор*. Наибольшие шансы выжить и закрепиться в табуне (популяции) имеют физически более мощные жеребята (способность

к самозащите, возможность победить соперников, способность к‘воспроизводству), обладающие большей скоростью (возможность спастись от хищников), выносливостью (способность преодолевать большие расстояния в поисках пищи). Тогда как слабые, больные жеребята (особи), а также жеребята, имеющие какие-либо отклонения (чересчур короткие или длинные ноги, слишком низкий или слишком высокий рост и т. д.), как правило, погибают. В то же время в ходе эволюции могут появляться особи с так называемыми «полезными» отклонениями от доминирующего в табуне (популяции) «стандарта» (генотипа), которые позволяют их носителям успешнее выживать в существующих условиях.

5. *Эволюционная смена форм*. Именно такие особи, как правило, выживают в результате схемы эволюции Дарвина и они постепенно, поколение за поколением становятся преобладающим (доминирующим) видом в данной популяции.

5.2. Модель эволюции Эйгена и Шустера, названная ими ***моделью гиперциклов***. Она моделирует условную стадию эволюции. М. Эйген описал добиологическую фазу эволюции, в ходе которой происходят процессы отбора, выражающие свойства вещества в особых системах реакций. Они известны как каталитические циклы. М. Эйген отмечал, что в далеких от равновесия биохимических системах каталитические реакции объединяются, формируя сложные сети, в которых могут содержаться и замкнутые циклы. М. Эйген установил, что в условиях достаточного времени и непрерывного потока энергии каталитические циклы сцепляются, образуя замкнутые циклы, в которых ферменты, созданные в одном цикле, являются катализатором в последующем цикле. Он назвал гиперциклами те петли, в которых каждый узел представляет собой каталитический цикл. На рис. 1.4. показана условная упрощенная схема модели гиперциклов.

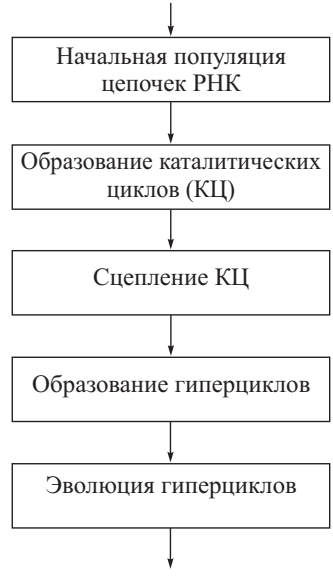


Рис. 1.4. Условная схема эволюции на основе гиперциклов

Гиперциклы способны к самовоспроизведению и коррекции ошибок при воспроизведении. Они могут хранить и передавать информацию. Следовательно, гиперциклы самоорганизуются, самовоспроизводятся и эволюционируют. Простейшая модель гиперциклов согласно Г. Хакену описывает автокаталитическое размножение биомолекул. Опишем модель Хакена. Пусть имеются два типа молекул: A и B. Все молекулы размножаются

путем автокатализа. Основная особенность состоит в том, что молекулы типа A выступают катализатором при размножении молекул типа B, и наоборот.

На рис. 1.5 приведен пример простейшего гиперцикла Эйгена, построенный Г. Хакеном. Здесь ИД — исходные данные, т. е. молекулы исходных веществ, поставляющих молекулы каждого типа, A и B. Очевидно, что гиперцикл может содержать молекулы 3, 4, ... типов. Гиперциклы подвергаются мутациям и молекулы могут размножаться путем автокатализа, вступать в конкурентную борьбу между собой.

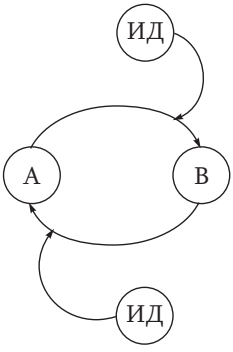


Рис. 1.5. Пример простейшего гиперцикла Эйгена

Приведем теперь модель эволюции Эйгена, которую он назвал ***моделью квазивидов***. Эта модель описывает процесс возникновения простейших макромолекул, кодирующих наследственную информацию. В модели рассматривается эволюция популяции цепочек РНК, которые могут размножаться путем редупликации. На основе такой эволюции создается квазивид, т. е. распределение цепочек РНК в окрестности некоторой «оптимальной РНК». На рис. 1.6 приведена условная схема модели эволюции квазивидов.

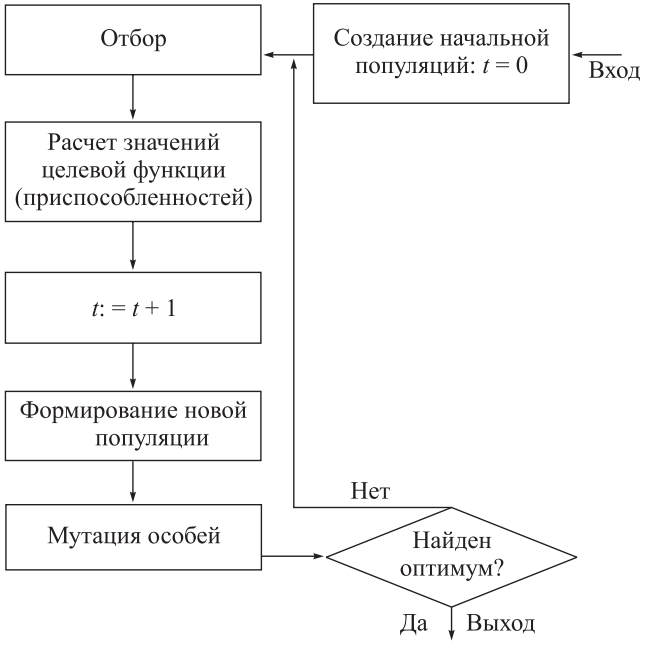


Рис. 1.6. Условная схема эволюции в модели квазивидов

Отдельная особь здесь задается геномом. Он представляет собой цепочку из n символов. Длина последовательности n и размер популяции не меняются в процессе эволюции. Геномы определяют приспособленность организмов. В простейшем случае оптимальная последовательность имеет максимальную приспособленность (целевую функцию).

Эволюционный процесс состоит из последовательности поколений $t = 0, 1, 2, \ldots$. В результате эволюции отбирается не отдельный вид, а квазивид — распределение видов. В детерминированном случае можно оценить распределение особей в квазивиде, скорость сходимости к этому распределению и условия, при которых определяется оптимальная особь. Согласно Г. Хакену модель квазивидов — это базовая модель эволюции информационных последовательностей.

5.3. ***Модель эволюции Ламарка***. Она основана на предположении, что характеристики, приобретенные особью (организмом) в течение жизни, наследуются его потомками. Эти изменения, как утверждал Ж. Ламарк, вызываются прямым влиянием внешней среды, упражнением органов и наследованием приобретенных при жизни признаков. Он объясняет одну из особенностей эволюции органического мира приспособляемостью. Прогрессивную эволюцию, появление форм, более сложных и совершенных, он объяснял «законом градаций» — стремлением живых существ усложнять свою структуру. Согласно Ж. Ламарку виды эволюционируют, приспособляясь и усложняясь, потому что у них существуют свойства — приспосабливаться и усложняться. Причины направленных изменений объясняются различно, но их можно свести к двум:

- направленное влияние внешней среды;
- способность самого организма.

Согласно Ж. Ламарку предполагается, что живые организмы способны сами находить верное решение, как себя улучшить, и, более того, сами же способны свое решение осуществлять. Направленная изменчивость здесь не причина, а всегда результат эволюционного процесса. Эта эволюция является концепцией искусственной эволюции, применимой в науке и технике, но не используемой в биологии. Авторы предлагают использовать некоторые принципы этого типа эволюции для решения оптимизационных задач. На рис. 1.7 приведена условная упрощенная схема модели эволюции Ламарка.

На основе данной схемы можно составить примерный алгоритм модели эволюции по Ламарку. Рассмотрим схему эволюции на уже знакомом нам сообществе особей (табун лошадей) из предыдущего примера.

1–2. *Популяция* + *Наследственность*. Как видно из схемы, первые два пункта у нас остались неизменными. Так же существует табун лошадей на некоторой территории обитания. Каждый год в табуне появляется потомство.

3. *Внешняя среда*. Происходит воздействие внешних факторов в виде сильной засухи в степях, где обитает наш табун. Трава в степи выгорела от засухи и табун в поисках пищи вынужден мигрировать на соседние территории, являющиеся предгорьями. На этих территориях более мягкий климат и трава все еще сохранилась.
4. *Приспособляемость*. В условиях изменившейся среды обитания (предгорья) популяции (табуна) более предпочтительные шансы на добычу корма и, следовательно, на выживание имеют коротконогие, невысокие особи, которым гораздо удобнее передвигаться в условиях пересеченной гористой местности.

5–6. *Отбор + Эволюционная смена форм*. Соответственно особи именно такого типа будут иметь преобладающие шансы на выживание и, следовательно, на закрепление своего фенотипа в популяции. Таким образом, если условия внешней среды останутся неизменными достаточно долго и данный табун продолжит обитать в данной местности, то после нескольких поколений указанный тип станет доминирующим в данной популяции. В свою очередь, высокие длинноногие лошади, скорее всего, обречены на вымирание в изменившихся условиях обитания.

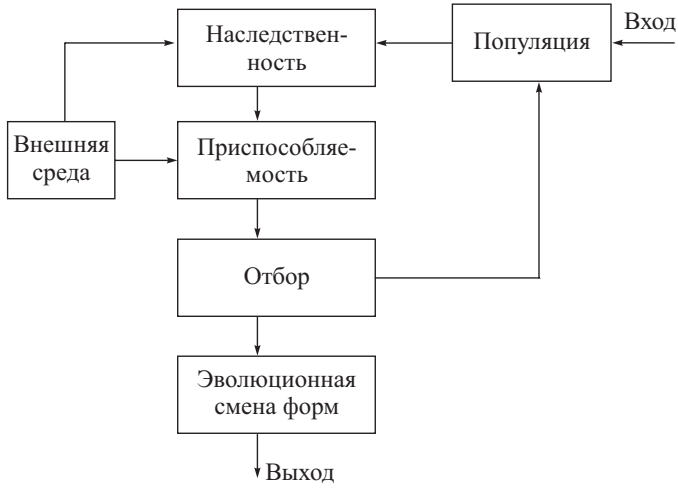


Рис. 1.7. Условная упрощенная схема модели эволюции Ламарка

Данная модель не получила применения в биологии. Авторы считают, что ее применение является полезным при решении технических задач, когда популяция имеет сходимость в области локального оптимума.

5.4. *Модель эволюции де Фриза*. В ее основе лежит моделирование социальных и географических катастроф, приводящих к резкому изменению видов и популяций. Эволюция, таким образом, представля-

ет собой последовательность скачков в развитии популяции без предварительного накопления количественных изменений в эволюционных процессах. Такой механизм эволюции иногда называют эволюцией катастроф. Он проявляется, ориентировочно, один раз в несколько тысяч поколений. Основная идея его состоит во внесении глобальных изменений в генофонд на момент катастрофы. На рис. 1.8 приведена условная упрощенная схема модели эволюции де Фриза.

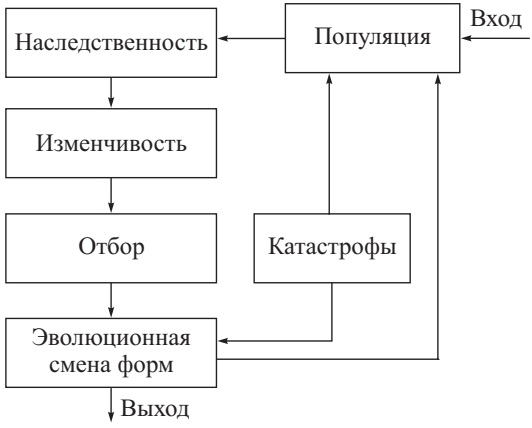


Рис. 1.8. Условная упрощенная схема модели эволюции де Фриза

Тогда алгоритм эволюции по де Фризу можно представить следующим образом. Пусть существует известный нам табун лошадей из предыдущих примеров. В ходе эволюции данной популяции происходит постепенная смена поколений, причем происходящие изменения генотипа популяции носят регулярный постепенный характер. Однако на некотором шаге эволюции данная популяция случайным образом подвергается катастрофическому воздействию внешней среды, которое приводит к значительному сокращению (или вымиранию) популяции и вызывает кардинальное изменение генотипа. В нашем примере таким фактором может быть большая засуха, в результате которой были уничтожены кормовые угодья. Вследствие случившегося голода выживает лишь незначительное количество особей из рассматриваемого табуна. Эти пережившие катастрофу особи составят впоследствии новую популяцию (табун) особей, с новыми качествами (рост, цвет и т. д.), которая заменит старую популяцию, и процесс эволюции будет продолжаться.

5.5. ***Модель Поппера*** — это условная структура, реализующая иерархическую систему гибких механизмов управления, в которых мутация интерпретируется как метод случайных проб и ошибок, а отбор — как один из способов управления с помощью устранения ошибок при взаимодействии с внешней средой. К. Поппер интерпретировал эволюцию Дарвина в виде триады: `дедуктивизм – отбор – устранение`

ошибок. Эволюция Поппера излагается в виде двенадцати тезисов. Основными из них являются:

- проблемы эволюции всегда решаются методом проб и ошибок;
- устранение ошибок может осуществляться либо путем полного устранения неудачных форм, либо в виде эволюции механизмов управления;
- популяция использует тот механизм управления, который выработался в процессе эволюции;
- популяция является пробным решением, анализируемым в процессе эволюции, выбирающим окружающую среду и преобразующим ее;
- эволюционная последовательность событий представляется в виде последовательности $F_1 \to TS \to EE \to F_2$, где F_1 — исходная проблема, TS — пробные решения, EE — устранение ошибок, F_2 — новая проблема.

В отличие от эволюции Дарвина, где существует одна проблема — «выживание сильнейших», в эволюции Поппера существуют и другие проблемы: воспроизводство, избавление от лишнего потомства и т. п. Согласно К. Попперу естественные системы исследуют окружающую среду и активно получают из нее информацию. Процесс выбора лучшей индивидуальности в данной эволюции может являться процессом отбора (селекции), а отбор из некоторого множества случайных событий не обязан быть случайным. На рис. 1.9 приведена условная упрощенная схема модели эволюции Поппера.

Обобщенный алгоритм функционирования модели эволюции Поппера объясним на примере действий ребенка. Ведь именно таким

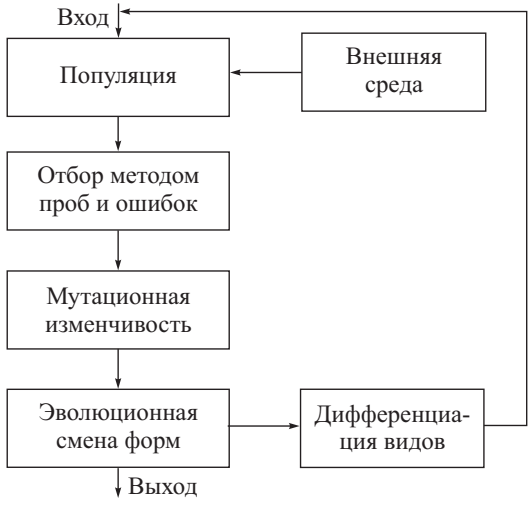


Рис. 1.9. Условная упрощенная схема модели эволюции Поппера

способом — методом проб и ошибок — постигают окружающий мир наши дети. Ребенок, сталкиваясь в жизни с той или иной незнакомой вещью или предметом, с помощью органов чувств собирает информацию о нем, пытается применить этот предмет для удовлетворения собственных потребностей. В случае, если исследуемый предмет ему не нравится или доставляет неудобства, он либо отказывается от его использования, либо пытается приспособить его для других целей. В любом случае в процессе реализации метода «проб и ошибок» ребенок получает и накапливает информацию о различных объектах окружающего мира, которую он затем использует для функционирования.

5.6. *М. Кимура* предложил модель нейтральной эволюции с нейтральным отбором. По его теории на генетическую изменчивость исходно влияют мутации, обуславливающие изменчивость и генетический дрейф, исключающий ее в отсутствии дифференцирующего отбора. Теория нейтральности предполагает, что большая часть молекулярных вариантов имеет равную приспособленность друг относительно друга. Изменчивость здесь поддерживается балансирующим отбором. Идея такого отбора следующая.

Пусть задана популяция, состоящая из «больших» и «маленьких» особей. Тогда эволюция заключается в реализации последовательностей поколений. Процесс реализации поколения состоит из двух шагов. На первом шаге дублируются все особи: большие имеют два больших потомка, маленькие — имеют два маленьких потомка. На втором шаге из популяции случайным образом удаляется ровно половина особей с равной вероятностью для больших и маленьких. Рассматриваемый процесс всегда сходится к одному из поглощающих состояний (все особи большие или все маленькие). При большом размере популяции $N \geqslant 1000$ характерное число поколений T_N, требуемое для сходимости к какому-либо из поглощающих состояний, равно $2N$. Данный эволюционный процесс чисто нейтральный, в результате эволюции выбирается только один вид. На рис. 1.10 приведена условная упрощенная схема модели нейтральной эволюции. Отметим, что блок случайной ре-

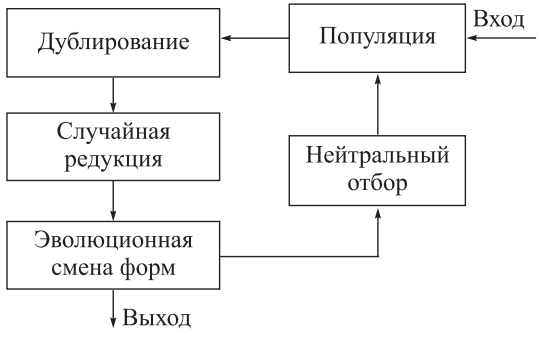


Рис. 1.10. Условная упрощенная схема модели нейтральной эволюции

дукции позволяет с равной вероятностью удалять из популяции ровно половину особей.

Механизм работы модели нейтральной эволюции Кимуры можно пояснить на примере знакомого нам табуна лошадей. Пусть существует некий табун лошадей (популяция). В этом табуне имеется две большие группы особей (лошадей), отличающиеся друг от друга окраской (вороные и гнедые). Тогда согласно данной модели в результате эволюции по прошествии некоторого определенного числа поколений в табуне останется только одна доминирующая группа особей (лошадей) одинаковой окраски.

5.7. *Модель синтетической эволюции*, описанная Н. Дубининым, представляет интеграцию различных моделей эволюций, в том числе Ч. Дарвина, Ж. Ламарка и Г. де Фриза. Условная упрощенная модифицированная схема модели синтетической теории эволюции показана на рис. 1.11.

Основные положения синтетической теории эволюции согласно Н. Дубинину следующие:

- Эволюция невозможна без адаптации организмов к условиям внешней среды. Фактором, формирующим приспособленность строения и функций организмов, выступает естественный отбор, который использует случайные мутации и рекомбинации.

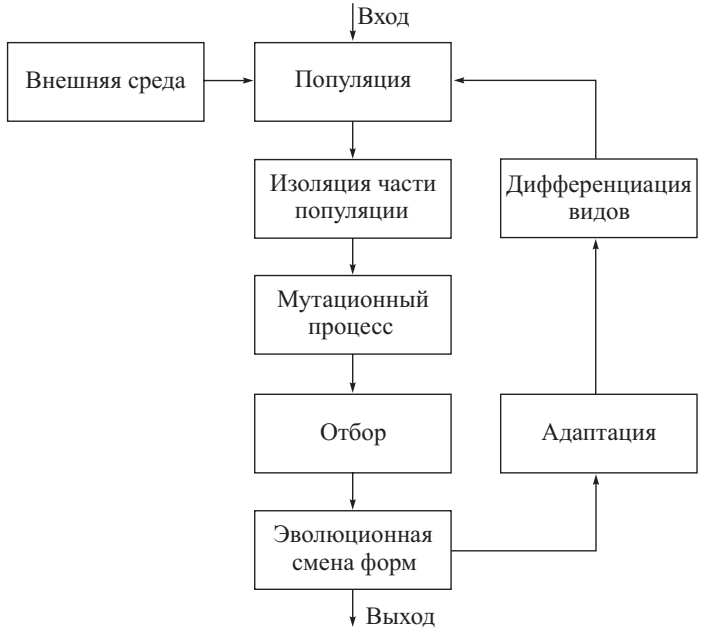


Рис. 1.11. Условная упрощенная модифицированная схема модели синтетической теории эволюции

- Естественный отбор, опираясь на процессы преобразования генетики популяций, создает сложные генетические системы. Их модификации закрепляются стабилизирующим отбором.
- В популяциях наследственная изменчивость имеет массовый характер. Появление специальных мутаций свойственно лишь отдельным особям.
- Наиболее приспособленные особи оставляют большее количество потомков.
- Специальные виды эволюций идут путем фиксации нейтральных мутаций на основе стохастического процесса.
- Реальным полем эволюции являются интегрированные генетические системы.
- Противоречия между случайным характером наследственной изменчивости и требованиями отбора определяют уникальность видовых генетических систем и видовых фенотипов.

Процессы, происходящие при реализации синтетической теории эволюции по Н. Дубинину, включают случайные, периодические и скачкообразные колебания, а также колебания численности отдельных популяций, обусловленные процессом миграции.

Отметим, что признание единства факторов эволюции в виде наследственности, изменчивости и естественного отбора не исключает существование разных форм эволюций. Н. Дубинин выделяет четыре основные формы осуществления внутренне единого эволюционного процесса:

- микроэволюция (процессы внутривидовой эволюции);
- эволюция на основе фазы нарастающего эволюционного усовершенствования;
- эволюция на основе переломных моментов;
- эволюция на основе интеграционных особенностей в организации естественных систем.

5.8. Эволюционный процесс связан с двумя типами адаптаций. Один тип адаптации основан на выработке приспособлений к условиям внешней среды, в которых вид существует в настоящее время. Другой тип адаптации связан с выработкой таких особенностей в структуре, которые должны обеспечить его соревнование с другими видами во времени.

Основная задача синтетической теории эволюции — определение природы противоречий или постепенной эволюции, т. е. разных форм противоречий между наследственностью и постоянно меняющимися потребностями в приспособлениях.

Кардинальное положение синтетической теории эволюции — признание стохастичности процессов мутаций и больших резервов рекомбинационной изменчивости. Условия внешней среды — не только факторы исключения неприспособленных особей, но и особенности, формирующие синтетическую теорию эволюции.

В этой связи авторы считают важным объединение всех видов и моделей эволюций. На рис. 1.12 приведем условную упрощенную интегрированную схему эволюции. Отметим, что блоки 1–5 соответствуют схемам моделей эволюций Дарвина, Ламарка, де Фриза, Поппера и Кимуры ***соответственно***. Основным этапом в каждой модели эволюции является анализ популяции, ее преобразование тем или иным способом и эволюционная смена форм.

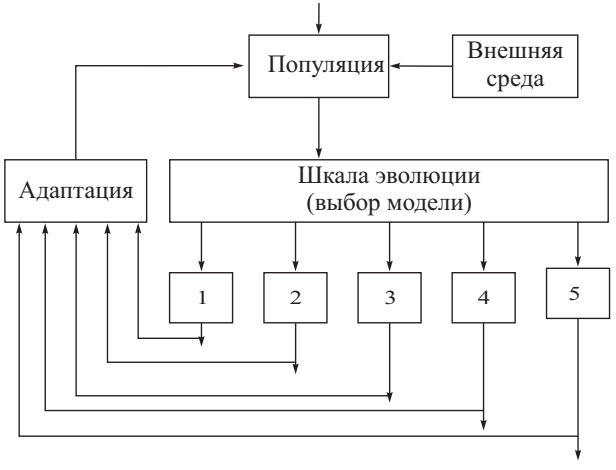


Рис. 1.12. Условная упрощенная интегрированная схема эволюции

Пусковой механизм эволюции функционирует в результате совместного действия эволюционных факторов в пределах популяции. В результате действия эволюционных сил в каждой популяции возникали элементарные эволюционные изменения. Со временем некоторые из них суммируются и ведут к возникновению новых приспособлений, что и лежит в основе видообразования.

1.6. Выводы

На основе приведенного краткого обзора можно заключить, что важными для решения различных практических задач являются:

- модели эволюций;
- механизм передачи наследственной информации;
- способы построения популяций;
- критерии эволюции;
- кроссинговер;
- мутация;
- инверсия;
- делеция;
- дупликация;

- транслокация;
- транспозиция;
- селекция (отбор);
- адаптация;
- синтетическая теория эволюции.

Основополагающей в эволюции является роль кроссинговера. Она состоит в создании из имеющегося генетического материала желаемой комбинации функций в одном решении. Задача мутации состоит в том, чтобы вывести качественно не улучшающееся посредством кроссинговера решение на новый уровень. Роль селекции — ускорение получения качественного решения; такая же роль принадлежит инверсии, делеции, дупликации, транслокации и транспозиции и др.

В последнее время проявляются тенденции использования естественных аналогов при создании моделей, технологий, методик, алгоритмов для решения задач, стоящих перед наукой и техникой. В подавляющем большинстве случаев использование естественных аналогов дает положительные результаты. Это объясняется тем, что аналог, взятый из природы, совершенствовался в течение многих лет эволюции и имеет на данный момент оптимальную структуру.

Один эволюционный процесс отличается от другого видом объектов, способом размножения, числом объектов в популяции, числом поколений и критериями выживания. При математической интерпретации его основная идея состоит в следующем. Производится перебор по всей популяции, в результате чего возникают объекты двух поколений — родители и потомки. При этом метод эволюции превращается в алгоритм смены поколений, в котором потомок становится родителем только в следующей генерации.

Одним из подходов такой модернизации является использование теории эволюционного моделирования, генетических алгоритмов, адаптационных принципов управления во взаимодействии с внешней средой. Адаптация позволяет накапливать и использовать информацию, достигать ее оптимального состояния при первоначальной неопределенности и изменяющихся внешних условиях.

Рассмотренные выше механизмы эволюции предлагается использовать в качестве случайного, направленного и комбинированного перебора при решении оптимизационных задач.

1.7. Контрольные вопросы

1. Кто считается основоположником генетики?
2. Назовите основные законы Менделя.
3. В чем заключается гипотеза Менделя?
4. Какой, по определению Менделя, является наследственность: непрерывной или дискретной?
5. Приведите определение гомологичной хромосомы.
6. Изложите гипотезу о материальных задатках.

7. В чем заключается роль хромосомы в механизме формирования наследственности?
8. Опишите процесс деления клеток.
9. Дайте определение кроссинговера и поясните его реализацию.
10. Приведите различные схемы кроссинговера.
11. Приведите общий принцип существования организмов по Дарвину.
12. Влияет ли, по мнению Ламарка, окружающая среда на развитие (изменение) организмов?
13. Кто впервые ввел в науку термины ген, генотип, фенотип, аллель, локус? Дайте их определение и опишите функциональное значение.
14. Дайте определение вида и популяции.
15. Какие факторы меняют генетический состав природной популяции?
16. Перечислите и поясните основные положения теории Моргана.
17. Приведите основные постулаты молекулярной генетики.
18. Назовите типы перераспределения наследственных факторов.
19. Дайте определение рекомбинации.
20. Какая связь между регулярной, специфической и нереальной рекомбинациями?
21. Приведите определение и дайте классификацию генетической изменчивости.
22. Дайте определение и поясните суть мутации.
23. Каковы сущности комбинативной и мутационной изменчивостей и их отличия?
24. Опишите систему классификации мутаций.
25. В чем заключается естественная роль мутации?
26. Какова суть генной мутации?
27. Опишите схему мутации, предложенную Г. де Фризом.
28. Приведите определение и основные виды внутрихромосомных перестроек.
29. Дайте определение операций делеции и инверсии.
30. Дайте определение, опишите типы операций транслокации и приведите их примеры.
31. Дайте определение и опишите виды селекции.
32. Дайте определение естественного отбора.
33. Дайте характеристику основных форм отбора по Дарвину.
34. Раскройте основные положения эволюционной теории Дарвина.
35. Что является движущей силой эволюции по Дарвину?
36. Приведите основные идеи теории наследования.
37. Каковы правила отбора наилучших форм организмов?
38. В чем заключаются отличия массового отбора от индивидуального?
39. Дайте определение инбридинга и приведите примеры его применения.
40. Опишите механизмы передачи наследственной информации.
41. Что такое локусы и как они связаны с генами и аллелями?
42. Поясните закон Харди–Вайнберга.
43. Опишите модели эволюции квазивидов и гиперциклов.

44. Что является движущим фактором эволюции Ламарка?
45. Опишите модель эволюции де Фриза.
46. Приведите основные тезисы эволюции Поппера.
47. Опишите идею нейтральной эволюции Кимуры.
48. Какова роль изоляции в процессе эволюции?
49. Какова роль адаптации в эволюционном процессе?
50. В чем заключаются основные идеи синтетической эволюции Дубинина?

1.8. Упражнения

1. Постройте схему выполнения одноточечного кроссинговера.
2. Опишите алгоритм реализации одноточечного кроссинговера.
3. Приведите пример схемы выполнения двухточечного кроссинговера.
4. Постройте алгоритм реализации двухточечного кроссинговера.
5. Приведите пример схемы выполнения модифицированного (многоточечного) кроссинговера.
6. Постройте алгоритм выполнения модифицированного кроссинговера.
7. Для некоторой случайно заданной популяции, где число аллелей 10, а число генов — 5, подсчитайте число возможных генотипов.
8. Приведите пример схемы реализации дупликации.
9. Опишите алгоритм выполнения дупликации.
10. Приведите пример схемы реализации делеции.
11. Сформулируйте алгоритм выполнения делеции.
12. Приведите пример схемы реализации перецентрической инверсии.
13. Сформулируйте алгоритм выполнения перецентрической инверсии.
14. Приведите пример схемы реализации парацентрической инверсии.
15. Сформулируйте алгоритм выполнения парацентрической инверсии.
16. Приведите пример схемы реализации реципрокной транслокации.
17. Опишите алгоритм выполнения реципрокной транслокации.
18. Приведите пример схемы реализации нереципрокной транслокации (транспозиции).
19. Сформулируйте алгоритм выполнения нереципрокной транслокации (транспозиции).
20. Приведите пример схемы реализации центрической (робертсоновской) транслокации.
21. Опишите алгоритм выполнения центрической (робертсоновской) транслокации.
22. Пусть численность популяции равна 100 особям ($N_0 = 100$), скорость замещения равна 3. Определите предполагаемую численность популяции через 10 и 100 поколений.
23. Постройте алгоритм реализации модели эволюции Дарвина.
24. Постройте модель эволюции гиперциклов для 3-х элементов.
25. Постройте алгоритм реализации модели эволюции Ламарка.
26. Постройте алгоритм реализации модели эволюции де Фриза.

27. Постройте алгоритм реализации модели эволюции Поппера.
28. Постройте алгоритм реализации модели эволюции Кимуры.
29. Постройте алгоритм реализации синтетической модели эволюции.

Глоссарий к разделу 1

Адаптация — процесс, а также результат приспособления строения и функций организмов к условиям внешней среды.

Аллель — различные формы гена.

Ген — единица наследственной информации, неделимая в функциональном отношении, которая передается от родителей к потомкам. Она рассматривается как участок молекулы ДНК, кодирующий синтез одной макромолекулы или выполняющий какую-либо другую элементарную функцию.

Генетика — биологическая наука, изучающая наследственность и изменчивость живых организмов.

Генетическая изменчивость — изменения, произошедшие в структуре генотипа и передаваемые по наследству.

Геном — копия всех генов организма (комплекс генов, содержащихся в наборе хромосом одного организма).

Генотип — генетическая конструкция организма. Совокупность всех генов, находящихся в хромосомах организма.

Генные мутации — принцип преобразования хромосом, в процессе реализации которого участвует, как правило, один или несколько генов. При этом один ген (или их последовательность) может превратиться в другой, может выпасть либо дублироваться, а группа генов может развернуться на 180°.

Гетерозигота — особь, в которой все аллели отличаются.

Гомозигота — особь, в которой все аллели данного гена одинаковы.

Гомологичные хромосомы — пары одинаковых хромосом, несущие гены одинаковых признаков.

Движущие силы эволюции (по определению Ч. Дарвина) — неопределенная изменчивость; борьба за существование; естественный отбор.

Дискретность — свойство генетического материала, позволяющее выделять в нем отдельные фрагменты, контролирующие те или иные функции.

Естественный отбор — общий принцип живой природы. Движущая сила эволюции. Процесс, направленный к повышению (или понижению) вероятности оставления потомства одной формой организмов по сравнению с другими.

Зигота — клетка с нормальным двойным набором хромосом, получившаяся в результате слияния ядер двух клеток.

Изменчивость — разнообразие признаков и свойств у особей и групп особей любой степени родства.

Инбридинг — скрещивание в популяции, где скрещивающиеся особи находятся в большем или меньшем родстве друг к другу.

Инверсии — повороты участка или всей хромосомы на 180°.

Кроссинговер — процесс сближения двух гомологичных хромосом (одна от отца, а другая — от матери) при созревании половых клеток и обмен частями.

Линейность — одномерность записи генетической информации. Она проявляется в определенной последовательности генов в пределах группы сцепления.

Локус — место нахождения конкретного гена на хромосоме.

Метод «выживания сильнейших» — процесс, посредством которого особи некоторой популяции, имеющие более высокое функциональное значение, получают большую возможность для воспроизведения потомков, чем «слабые» особи.

Механизмы передачи наследственной информации: бесполое размножение, основано на равном делении ядер клеток, при котором происходит простое копирование наследственной информации без какого-либо изменения; ***половое размножение***, основано на редукционном и последующем равном делениях ядер клеток двух родителей, в результате чего потомок получает копии наследственной информации обоих родителей.

Модель гиперциклов моделирует условную стадию эволюции, в ходе которой происходят процессы отбора, выражающие свойства вещества в особых системах реакций.

Модель квазивидов описывает процесс возникновения простейших макромолекул, кодирующих наследственную информацию.

Модель синтетической эволюции Дубинина представляет собой интеграцию различных моделей эволюций.

Модель эволюции Дарвина — условная структура, реализующая процесс, когда особи некоторой популяции, имеющие высокое функциональное значение, получают бóльшую возможность для воспроизведения потомков.

Модель эволюции Кимуры — эволюция с нейтральным отбором. Рассматриваемый процесс всегда сходится к одному из поглощающих состояний. В результате эволюции выбирается только один вид.

Модель эволюции Ламарка — основана на предположении, что характеристики, приобретенные особью в течение жизни, наследуются его потомками.

Модель эволюции Поппера — иерархическая структура, реализующая систему механизмов управления, в которых мутация интерпретируется как метод случайных проб и ошибок, а отбор — как один из способов управления при взаимодействии с внешней средой.

Модель эволюции де Фриза представляет собой последовательность скачков в развитии природы без предварительного накопления количественных изменений в эволюционных процессах.

Мутации — наследственные изменения отдельных генов.

Мутационная изменчивость — процесс изменения генетической структуры организма, его генотипа.

Непрерывность — свойство физического единства групп сцепления. Оно означает, что определенные комбинации генов совместно контролируют некоторую функцию.

Относительная стабильность генетического материала — способность к изменениям — проявляется в мутациях, что означает способность генов к воспроизведению и изменению с последующим воспроизведением измененных вариантов.

Популяция — это совокупность особей определенного вида, в течение большого числа поколений населяющих определенный участок пространства, внутри которого осуществляется случайное скрещивание.

Рекомбинация — перераспределение наследственных факторов.

Селекция — форма искусственного отбора, где эволюция направляется факторами внешней среды.

Транслокации — межхромосомные перестройки, при которых участок хромосомы перемещается (транслоцируется) на другое место.

Транспозиции — межхромосомные перестройки, при которых участок хромосомы изменяет свое положение или включается в другую хромосому без взаимного обмена.

Фенотип — совокупность внешних признаков, характеризующих организм.

Хромосомные мутации — конструкция, в процессе построения которой происходит изменение числа, размеров и организации хромосом. Существуют внутрихромосомные и межхромосомные перестройки.

Хромосомы — представляют собой нитевидные структуры, находящиеся в клеточном ядре. Моделью хромосомы является нить, на которую «словно бусинки» нанизаны гены.

Эволюция — процесс постепенного и непрерывного изменения форм организмов от одного состояния к другому.

Список литературы к разделу 1

1. *Алиханян А. В. и др.* Общая генетика. — М.: Наука, 1985.
2. Большая советская энциклопедия. Т. 29. — М.: Изд-во Сов. энцикл., 1978.
3. *Вернадский В. И.* Биосфера и ноосфера. — М.: Рольф, 2002.
4. *Дарвин Ч.* Происхождение видов путем естественного отбора. — М.: Тайдекс Ко, 2003.
5. *Дженкинс М.* 101 ключевая идея: Генетика. — М.: ФАИР-ПРЕСС, 2002.
6. *Дубинин Н. П.* Избранные труды. Т. 1. Проблемы гена и эволюции. — М.: Наука, 2000.
7. *Емельянов В. В.*, *Курейчик В. В.*, *Курейчик В. М.* Теория и практика эволюционного моделирования. — М.: Физматлит, 2003.
8. *Инге-Вечтомов С. Г.* Введение в молекулярную генетику. — М., 1982.
9. *Капра Ф.* Паутина жизни. Новое научное понимание живых систем. — К.: София; М.: ИД Гелиос, 2002.
10. *Кордюм В. А.* Эволюция и биосфера. — Киев: Наукова Думка, 1982.

11. *Майер Э.* Популяции, виды и эволюция. Пер. с англ. — М.: Мир, 1974.
12. *Приходченко Н. Н.*, *Шкурат Т. П.* Основы генетики человека. — Ростов-на-Дону: Феникс, 1997.
13. *Хакен Г.* Тайны природы. Синергетика: учение о взаимодействии. — М., Ижевск: Институт компьютерных исследований, 2003.
14. *Хедрик Ф.* Генетика популяций. — М.: Техносфера, 2003.
15. Эволюционная эпистемология и логика социальных наук: Карл Поппер и его критики // Сост. *Д. Г. Лахути*, *В. Н. Садовского*, *В. К. Финна*. — М.: Эдиториал УРСС, 2000.
16. *Эйген М.* Самоорганизация материи и эволюция биологических макромолекул. — М.: Мир, 1973.
17. *Эйген М.*, *Шустер П.* Гиперцикл. Принципы самоорганизации макромолекул. — М.: Мир, 1982.

Аналоги, взятые из природы, совершенствовались в течение многих лет эволюции и имеют совершенную структуру. Поэтому их использование при решении практических задач дает эффективные результаты

2. МЕТОДЫ ОПТИМИЗАЦИИ

Так как здание всего мира совершенно и выведено премудрым творцом, то в мире не происходит ничего, в чем бы не был виден смысл какого-нибудь максимума или минимума; поэтому нет никакого сомнения, что все явления мира с таким же успехом можно определить из причин конечных при помощи метода максимумов или минимумов, как из самих причин производящих.

Л. Эйлер

2.1. Постановка оптимизационных задач

1. ***Оптимальность*** — оценочное отражение субъективного свойства через некоторое количественное соотношение, т. е. количественное значение качества, которое желательно придать моделируемой задаче.

Моделирование — исследование явлений, процессов путем построения их моделей.

Модель — копия или аналог изучаемого явления или процесса, отражающая существенные свойства моделируемого объекта с точки зрения исследователя; приближенное описание изучаемого класса явлений внешнего мира.

2. Опишем понятие математической модели рассматриваемых объектов оптимизации. Количественная оценка существенных свойств модели, режимов ее работы или внешней среды, в которой она функционирует, определяется значениями множества переменных. Входные переменные представляются вектором $X = (x_1, x_2, \ldots, x_n)$. Они характеризуют свойства внешней по отношению к модели среды, оказывающей влияние на ее функционирование. Внутренние переменные (переменные состояния) — величины, которые характеризуют состояние отдельных элементов модели. Обозначим их вектором $Z = (z_1, z_2, \ldots, z_r)$. Величины, характеризующие свойства модели в целом и определяющие степень выполнения моделью своего назначения, называют выходными

переменными. Их обозначают вектором $Y = (y_1, y_2, \ldots, y_m)$. Векторы X, Y, Z являются функциями времени.

Выражения, отображающие зависимость между входными и выходными переменными, а также переменными состояния, называют ***математической моделью*** решаемой задачи:

$$\begin{cases} y_1 = y_1\,(x_1, \ldots, x_n;\ z_1, \ldots, z_r)\,, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_m = y_m\,(x_1, \ldots, x_n;\ z_1, \ldots, z_r)\,. \end{cases}$$

Эту систему можно представить в форме

$$Y = F(X, Z). \tag{2.1}$$

Выражение (2.1) представляет собой отображение (или соответствие) между двумя множествами переменных X, Z, которое можно получить на основе одной из моделей. Таким образом, под ***математической моделью*** будем понимать зависимость F, которая в общем случае может быть некоторым алгоритмом, оператором, функцией, правилом, инструкцией и т. п. Она позволяет определить значения выходных переменных, не обращаясь к реальному объекту (системе), для которого решается задача.

Условно модель F можно представить как ее структуру St и вектор параметров $C = (c_1, c_2, \ldots, c_k)$:

$$F = \langle St, C \rangle\,.$$

Создание математической модели осуществляется в два этапа:

— структурного синтеза, когда определяется структура St, т. е. вид элементов, из которых состоит модель и отношения между ними;

— параметрического синтеза, связанного с определением численных значений параметров модели $C = (c_1, c_2, \ldots, c_k)$.

По известной структуре описания задачи и значениям векторов X и Z создают физические или математические модели F и на их основе определяют оценки значений вектора Y. Процедуры анализа систем обычно выполняют методом моделирования.

Функционально математические модели обычно отражают свойства исследуемой системы, связанные с различными процессами ее функционирования. Такие модели чаще всего представляют в виде систем уравнений, описывающих фазовые параметры, внутренние, внешние и выходные переменные.

3. Опишем теперь требования, предъявляемые к математическим моделям. Основными являются адекватность, точность, степень универсальности и экономичность.

3.1. ***Адекватность математической модели*** — соответствие модели решаемой задаче или процессу принятия решений. Причем адекватность рассматривают по тем свойствам модели, которые для лица, принимающего решение (ЛПР), являются наиболее важными в данный

момент времени. Под адекватностью математической модели также понимают способность отображать заданные свойства модели с текущей погрешностью $\varepsilon_{\text{Т}}$, не выше заданной погрешности $\varepsilon_{\text{З}}$. Математическая модель называется адекватной по вектору Y, если погрешность расчета на ее основе значений выходных параметров $y_i \in Y$ не превышает заданных. Адекватность математической модели обычно рассматривают в ограниченной области изменения входных переменных. Эту область называют областью адекватности математической модели. В ней выполняется неравенство $|\varepsilon_i| \leqslant \varepsilon_{i,g}$, где ε_i — относительная погрешность определения переменной y_i, возникающая из-за приближенного характера математической модели; $\varepsilon_{i,g}$ — допустимая погрешность ($\varepsilon_{i,g} \geqslant 0$).

3.2. Отметим, что любая математическая модель описывает лишь некоторое подмножество свойств задачи. Поэтому ***точность математической модели*** определяется как степень совпадения значений переменных реального объекта и значений тех же переменных, полученных на основе исследуемой математической модели. При определении точности математической модели важно определять погрешности моделей. Пусть, как и выше, $Y = (y_1, y_2, \ldots, y_m)$ — вектор выходных переменных; $y_{i,\text{э}}$ — эталонное значение выходной переменной, определенное на основе экспертных оценок; $y_{i,\text{мм}}$ — значение i-й выходной переменной, рассчитанное на модели. Тогда относительную погрешность ε_i при расчете выходной переменной определяют следующим образом:

$$\varepsilon_i = (y_{i,\text{мм}} - y_{i,\text{э}})/y_{i,\text{э}}.$$

3.3. ***Погрешность математической модели*** для всех значений переменных будет представлять вектор $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$, где n — число выходных переменных. Отметим, что значения погрешности, полученные таким образом, зависят от свойств математической модели, особенностей решаемых задач и достоверности экспертных оценок. В этой связи при анализе каждого класса практических задач необходимо пересматривать оценки точности математической модели.

3.4. Математическая модель — только частичное отображение реального объекта (или системы). Поэтому ***степень универсальности математической модели*** соответствует полноте учета в ней свойств реальной задачи. Например, модель конструкции схемы инженерных сетей в виде гиперграфа будет отображать только ее коммутационные свойства, не учитывая протекающих в ней химических, физических и информационных процессов. Усложняя модель, можно увеличивать степень приближения математической модели к реальной задаче.

3.5. ***Экономичность математической модели*** характеризуют затратами вычислительных ресурсов ЭВМ при ее реализации. Основными из таких ресурсов являются время $T_{\text{М}}$ и объем используемой памяти ЭВМ: $V_{\text{М}} = V_{\text{оп}} + V_{\text{вн}}$, где $V_{\text{оп}}$ — объем оперативной памяти; $V_{\text{вн}}$ — объем внешней памяти. Отметим, что так как $V_{\text{вн}} \gg V_{\text{оп}}$, то при

анализе затрат памяти в большинстве случаев оценку можно вести по объему внешней памяти $V_{\text{вн}}$. Математическая модель считается тем экономичнее, чем меньше значения $T_{\text{М}}$ и $V_{\text{М}}$. Величину $T_{\text{М}}$ определяют как усредненное число операций Ω, выполняемых при однократном обращении к модели. Величину $V_{\text{М}}$ определяют в основном числом B переменных математической модели. Сравнение математических моделей по экономичности состоит в сравнении значений Ω и B.

3.6. ***Верификация модели*** — проверка ее истинности, адекватности, сводится к сопоставлению расчетных результатов по модели с соответствующими данными действительности.

4. При построении математических моделей практических задач в основном используются неформальные и эвристические методы на основе нечетких множеств, нетрадиционных алгебр логики, эволюционного и имитационного моделирования.

Отметим, что в математической модели задачи выражение (2.1) устанавливает взаимно-однозначное соответствие между входными и выходными переменными и является ее детерминированным описанием. В реальной практике принятия решений параметры задачи часто имеют стохастический и нечеткий характер. Тогда параметры модели задаются как случайные величины, выбираемые согласно заданному закону распределения вероятностей.

Приведем неравенство для параметров задачи:

$$c_i^{\text{ном}} - \delta_i \leqslant c_i \leqslant c_i^{\text{ном}} + \delta_i, \quad i = 1, 2, \ldots, n, \tag{2.2}$$

где $c_i^{\text{ном}}$ — номинальное значение параметра модели c_i, которое считается его математическим ожиданием; δ_i — допуск на значение данного параметра, который определяется как половина интервала со значением $c_i^{\text{ном}}$ в центре.

Обозначим вектором $\varphi = (\varphi_1, \ldots, \varphi_r)$ множество параметров состояния $\varphi \subseteq C$, которые, являясь случайными величинами, влияют на значения выходных переменных. Кроме того, обозначим вектором $\psi = (\psi_1, \psi_2, \ldots, \psi_k)$ совокупность параметров состояния $\psi \subseteq C$, которые, являясь нечеткими величинами, также влияют на значения выходных переменных. При этом $\psi_i = (c_i, \mu_i)$, где μ_i — функция принадлежности элемента c_i к множеству ψ, т. е. $\mu_i : \psi \to [0, 1]$. Тогда математическое описание модели примет вид

$$Y = F(X, Z, \varphi, \psi). \tag{2.3}$$

5. Нечеткий алгоритм построения математического описания задачи можно представить в следующем виде:

1. Определить существенные с точки зрения ЛПР свойства задачи, которые необходимо отразить в ее математическом описании.
2. Разделить выбранные свойства на внешние, внутренние, неконтролируемые (случайные) и выходные переменные.

3. Выбрать математическую форму записи для выражения функциональных зависимостей между входными и выходными переменными.
4. Построить математическое описание в виде модели.
5. Реализовать математическую модель программно-аппаратно, что позволит по заданным входным переменным получать значения либо оценки выходных.
6. Провести математическое, имитационное или эволюционное моделирование для оценки численных значений параметров модели и проверки адекватности математической модели реальной задаче.
7. Оценить точность и сложность адекватности математического описания для обеспечения компромисса между ожидаемой точностью результатов математического моделирования и затратами вычислительных ресурсов.
8. Конец работы алгоритма.

6. Заметим, что для каждой задачи можно построить большое число математических моделей, которые отражают те или иные ее свойства. Область пространства управляемых переменных, в которых выполняется система накладываемых на их значения ограничений, называется областью поиска D_z. Эта система ограничений имеет вид

$$z_i^- \leqslant z_i \leqslant z_i^+, \quad i = 1, 2, \ldots, n,$$

здесь z_i^-, z_i^+ — соответственно нижнее и верхнее предельно допустимые значения для i-й управляемой переменной, а n — число управляемых переменных.

В процессе принятия решений стремятся выбирать значения вектора управляемых переменных, принадлежащего области поиска $z \in D_z$, таким образом, чтобы удовлетворить требования ЛПР. При формализации задачи удовлетворение требований сводится к выполнению системы соотношений между выходными переменными y_j и их предельно допустимыми по техническому заданию значениями y_j^- и y_j^+, называемыми ***условиями работоспособности***.

Область пространства управляемых переменных, в которой выполняются все условия работоспособности, называют ***областью работоспособности*** D_R. Тогда множество $D = D_Z \bigcap D_R$ называется областью допустимых решений.

Область D определяет множество всех работоспособных вариантов решений задачи, каждый из которых однозначно описывается вектором управляемых переменных $z \in D$. Одна из проблем принятия решений — реализовать процедуру выбора из работоспособных вариантов, одного или некоторого множества, предпочтительных с точки зрения ЛПР.

Принимая предварительное решение, ЛПР дихотомически разбивает область D на два подмножества: $D = D_1 \cup D_2$. В подмножество D_1 включаются отобранные по заданному критерию варианты решения, а в D_2 — все отклоненные варианты. Причем $D_1 \cap D_2 = \varnothing$, но на

каждой итерации принятия решения часть вариантов может переходить из D_1 в D_2, и наоборот.

Процесс построения математической модели нечетких процедур принятия решений состоит в следующем:

- выбрать параметры в качестве управляемых переменных z;
- построить области допустимых решений D;
- задать механизмы поиска, перебора и выбора правил генерации в области D работоспособных вариантов и определить подмножества D_1 и D_2.

7. Для сравнения вариантов принятия решений вводится критерий оптимальности Q — количественный показатель, характеризующий качество модели, с помощью которого осуществляется измерение одного наиболее важного для объекта свойства.

Под ***оптимизационной задачей*** понимается задача, в которой необходимо найти решение, в некотором смысле наилучшее или оптимальное. Отметим, что наилучшего решения во всех смыслах быть не может. Оно может быть признано оптимальным на основе ***критерия*** (меры оценки исследуемого явления) или ***целевой функции***. Существует большое количество оптимизационных задач. Они могут иметь различный характер. Однако постановка всех оптимизационных задач имеет много аналогий.

Во-первых, в оптимизационных задачах часто указывается исходное множество альтернативных вариантов решений. Из этого множества выбирается оптимальное решение. Это исходное множество решений называется пространством решений. В дальнейшем его будем обозначать через M.

Во-вторых, некоторые решения априорно отвергаются в качестве «плохих». Другими словами, в пространстве решений задаются ограничения, которым должны удовлетворять оптимальные решения. Эти ограничения позволяют выделить в пространстве решений M некоторое подмножество M' тех решений, которые удовлетворяют заданным ограничениям D.

В-третьих, указывается принцип сравнения любых двух допустимых решений с тем, чтобы можно было выяснить, какое из них лучше. Как правило, этот способ сравнения задается с помощью ***критерия оптимальности*** (или функционала, функции качества, функции полезности и т. п.).

Согласно Л. Лопатникову ***критерий оптимальности*** — тот признак, по которому функционирование системы признается наилучшим из возможных вариантов ее функционирования. Другими словами, критерий оптимальности — показатель, выражающий предельную меру экономического эффекта принимаемого решения для сравнительной оценки возможных альтернатив и выбора наилучшего из них. Например, это может быть максимум или минимум стоимости, кратчайший путь и т. п. Критерий оптимальности носит количественный характер

для того, чтобы качественный признак решения, выражаемый нечетким понятием «лучше–хуже», переводить в количественное определение «больше–меньше». Критерию оптимальности соответствует математическая форма — целевая функция. Возможным выражением критерия оптимальности является шкала оценок полезности, ранжирования, предпочтений и т. п.

Обозначим критерий следующим образом: $Q : M' \to R$, где R — множество неотрицательных вещественных чисел.

Зная функцию Q, можно реализовать процедуру сравнения вариантов решений. При этом решение $m \in M'$ лучше, чем решение $m' \in M'$, если $Q(m) < Q(m')$ (при поиске наименьшего значения Q). В этом случае говорят, что оптимизационная задача состоит в минимизации критерия Q, т. е. требуется найти такое допустимое решение $m'' \in M'$, что

$$Q(m'') = \min Q(m')/m' \in M'.$$

Модель оптимизационной задачи запишем в виде кортежа длины три: $\langle M, D, Q\rangle$, где M — пространство решений; D — ограничения, выделяющие в M область допустимых решений $M' \subseteq M$; $Q : M' \to R$ — критерий оптимизации. Требованием оптимизации является выражение: $Q(m) \to \min$ или $Q(m) \to \max$. Решение $m'' \in M'$, удовлетворяющее требованию оптимизации, называется оптимальным.

8. Целью оптимизационной задачи является выбор допустимого или оптимального решения из множества альтернатив для достижения поставленной цели. Оптимизационная задача должна удовлетворять двум основным требованиям: должны существовать как минимум два решения; надо знать, в каком смысле искомое решение должно быть наилучшим. Выбор задачи заканчивается ее содержательной постановкой. Модель описывает зависимость между исходными данными и искомыми величинами. Математическая модель оптимизационной задачи состоит из трех составляющих: целевой функции, ограничений, граничных условий. Часто классификацию оптимизационных задач проводят по трем основным принципам: область применения; содержание задачи; класс математических моделей.

9. Классификацию математических моделей проводят:
по элементам модели:
(a) детерминированные и случайные;
по искомым переменным:
(b) непрерывные и дискретные;
по зависимостям, описывающим целевую функцию, ограничениям и граничным условиям:
(c) линейные и нелинейные.

10. Комбинации элементов математических моделей приводят к различным классам оптимизационных задач. Наиболее простые — линейные оптимизационные задачи, однако на практике чаще всего встре-

чаются нелинейные. При решении оптимизационных задач возможна одна из двух постановок:

- при заданных условиях и ограничениях максимизировать получаемый результат;
- при заданном результате минимизировать используемые ресурсы.

Максимальное и минимальное значения целевой функции в оптимизационных задачах объединяют определением экстремума. Наибольшее или наименьшее значение функции без учета того, где находится такое значение — внутри заданного интервала или на его границе, называют не экстремумом, а оптимумом. Оптимум — более общее понятие, чем экстремум. Экстремум есть не у всех функций, а в оптимизационных задачах оптимум есть всегда. Оптимум может быть локальным и глобальным.

Глобальным максимумом (***минимумом***) называют такой максимум (минимум), который больше (меньше) всех остальных. В общем случае задача поиска глобального оптимума сводится к нахождению всех локальных оптимумов, если это возможно, и выбора из него наилучшего с точки зрения значения целевой функции.

Оптимальное решение $x^* \in D$ называется точкой локального минимума (локальным решением), если $\forall x \in d(x^*, \varepsilon)$, истинно высказывание $Q(x^*) \leqslant Q(x)$, здесь $d(x^*, \varepsilon)$ — ε-окрестность точки x. Оптимальное значение $x^* \in D$ называется точкой глобального минимума (глобальным решением), если ни в одной другой точке области допустимых решений функция $Q(x)$ не принимает меньших значений: $Q(x^*) \leqslant Q(x)\ \forall x \in D$. Следовательно, глобальный минимум — наименьший из всех локальных минимумов. Тогда ***квазиоптимальное решение*** — одно из множества решений, попадающих в локальный экстремум, близкий к глобальному.

2.2. Технологии локального поиска

Рассмотрим инструментальные средства, позволяющие находить локальный оптимум. К ним относятся методы одномерного поиска, градиентного поиска, а также статистические методы оптимизации. На рис. 2.1 показана одна из возможных классификаций таких стратегий поиска.

1. К ***одномерному поиску*** относят методы: пассивный, последовательный. ***Пассивный поиск*** заключается в случайном выборе пар на заданном отрезке для нахождения оптимума целевой функции. ***Последовательный поиск*** выполняется путем перебора значений целевой функции для нахождения оптимального значения. Опишем один из возможных методов одномерного поиска — алгоритм поиска из начальной точки.

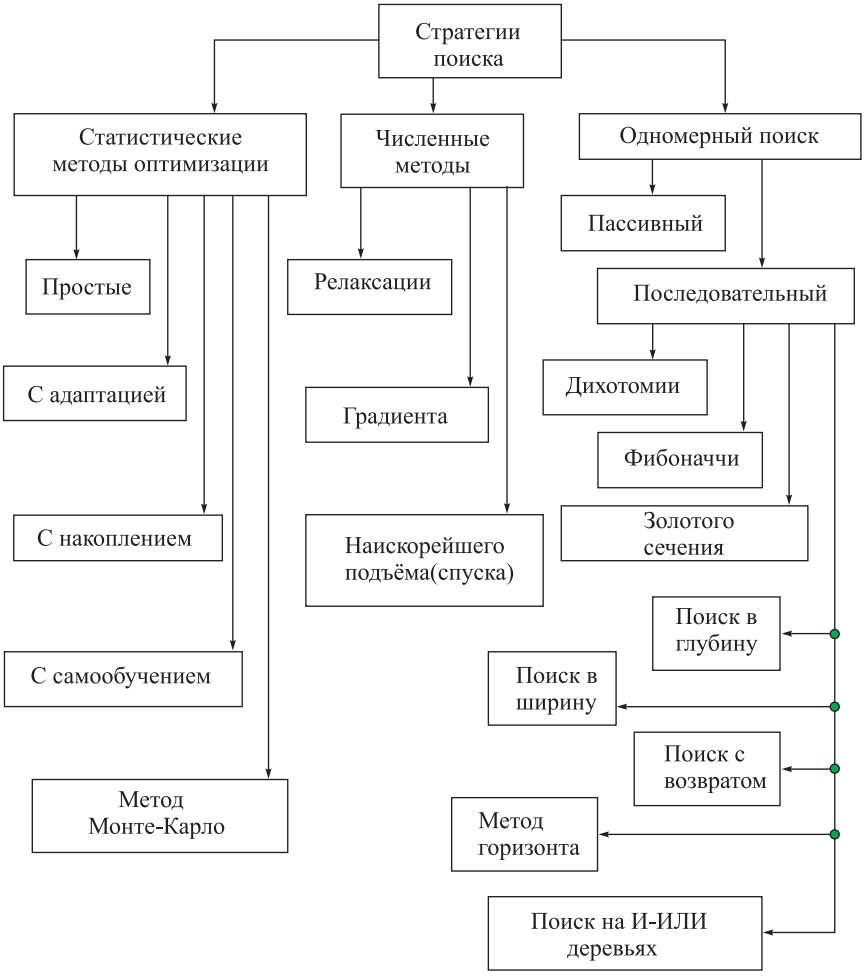


Рис. 2.1. Классификация стратегий поиска

1. Получить набор или одно альтернативное решение.
2. Построить график зависимости значений целевой функции от времени (или числа элементов модели).
3. Ввести начальную точку V_0, начальный шаг ΔV, точность ε, вычислить значение целевой функции $F(V_0)$.
4. Положить $V_1 = V_0 + \Delta V$, вычислить значение целевой функции $F(V_1)$.
5. При минимизации значения целевой функции, если значение $F(V_1) > F(V_2)$, то ΔV присвоить $= -\Delta V$ и перейти к п. 4.
6. Присвоить $\Delta V := 2\Delta V$.
7. Присвоить $\Delta V_2 := V_1 + \Delta V$, вычислить значение $F(V_2)$.

8. Если $F(V_2) < F(V_1)$, то присвоить $V_0 = V_1$, $V_1 = V_2$, $F(V_0) = F(V_1)$ и перейти к 6, иначе к 9.
9. Положить $a = V_0$, $b = V_2$, вывести $[a, b]$ — интервал неопределенности.
10. Конец работы алгоритма.

2. К последовательному поиску относятся методы: дихотомии, Фибоначчи, золотого сечения, поиска в глубину, в ширину, с возвратом, горизонта и поиск на И–ИЛИ деревьях.

Метод дихотомии реализуется за счет механизма обычного перебора возможных точек разрыва. Он аналогичен методу деления отрезка пополам для нахождения точки, в которой значение целевой функции имеет локальный оптимум.

Например, на отрезке $A = [a, h]$, представленном в виде набора точек

$$\overset{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8}{A : a|b|c|d|e|f|g|i|h},$$

возможно максимально 8 точек разрыва. Очевидно, точность (эффективность) такого процесса определяется количеством выбранных точек n, причем $n = L - 1$, где L — длина отрезка, т. е. количество выделенных дискретных точек. Дискретный интервал [1–8] обозначим L_0. Реализованный интервал назовем интервалом неопределенности $L_{\text{Н}}$.

Эффективность поиска будет составлять $2L_0/(n + 1)$, что приблизительно равно двум. Такой метод деления отрезка пополам для нахождения точки, в которой целевая функция имеет локальный оптимум, называется методом дихотомии. Равномерное распределение всех разрывов на интервале $[a, b]$ не является наилучшим. Эффективность поиска можно улучшить, если все разрывы проводить последовательно и попарно, анализируя результаты после каждой пары экспериментов. Наиболее эффективные результаты — такое расположение пары разрывов, при котором текущий интервал неопределенности сокращается практически вдвое. Итак, после первого разбиения получаем интервалы

$$L_0 = L_1 + L_2, \quad L_1 = \frac{1}{2}L_0 + \varepsilon, \quad L_2 = \frac{1}{2}L_0 - \varepsilon,$$

$$L_3 = \frac{1}{4}L_0 + \varepsilon, \quad L_4 = \frac{1}{4}L_0 - \varepsilon, \quad L_5 = \frac{1}{8}L_0 + \varepsilon, \quad L_6 = \frac{1}{8}L_0 - \varepsilon,$$

где $\varepsilon = 0, 1, 2, \ldots$ в зависимости от решения ЛПР.

Приведем укрупненный алгоритм метода дихотомии.

1. Получить набор или одно альтернативное решение.
2. Ввести границы интервала, параметр ε.
3. Разделить исходный отрезок пополам (при нечетном размере в любую часть берется ближайшее большее число).
4. Вблизи точки деления (по разные стороны с наименьшим интервалом) $\varepsilon = 0, 1, 2, \ldots$ найти точки с экстремальным значением целевой функции.

5. Каждую половину отрезка снова разделить пополам и процесс расчета продолжать по исходной схеме до тех пор, пока не будет получено наилучшее значение целевой функции или ранее не будет закончено деление на основе заданной метки.
6. Конец работы алгоритма.

Эффективность метода дихотомии экспоненциально растет с ростом n.

3. Оптимальную стратегию последовательного поиска дает ***метод Фибоначчи***. Сущность алгоритма состоит в том, что выборки, проводимые последовательно, располагаются так, чтобы для соседних выборок выполнилось условие

$$d_j = d_{j+1} + d_{j+2}, \tag{2.4}$$

здесь d_j — номер дискретной точки на интервале $[a, b]$. Для любой $(n-k)$-выборки справедливо

$$d_{n-k} = F_{k+1} d_n,$$

где F_k — числа Фибоначчи.

В алгоритме используется то свойство чисел ряда Фибоначчи, что очередной член ряда равен сумме двух предыдущих, кроме первого и второго $F_k = F_{k-1} + F_{k-2}$, где $F_0 = 0$, $F_1 = 1$, $k = 2, 3, \ldots$ (например, числа Фибоначчи 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...).

Вначале, исходя из заданной требуемой точности ε, определяется наименьшее по значению число из ряда Фибоначчи, удовлетворяющее неравенству

$$\Delta = (b-a)/F_n \leqslant \frac{\varepsilon}{2}.$$

Тогда $(b-a) = F_n \Delta$, поэтому весь интервал можно разделить на количество частей F_n, причем каждая часть будет иметь длину, меньшую ε ($\Delta < \varepsilon$), при этом $F_n = F_{n-1} + F_{n-2} = F_{n-2} + F_{n-3} + F_{n-2} = 2F_{n-2} + F_{n-3}$. Точки деления x_1 и x_2 определяются по формулам:

$$x_1 = a + F_{n-2}\Delta; \quad x_2 = x_1 + F_{n-3}\Delta = a + F_{n-1}\Delta.$$

В найденных точках проверяются значения целевой функции. Если лучшие значения целевой функции достигнуты в точке x_i, то в качестве интервала выбирается $[a, x_i]$, в противном случае $[x_i, b]$. Для продолжения поиска точки следующего замера следует расположить симметрично имеющемуся разбиению. Далее поиск осуществляется аналогично. Точка искомого экстремума определяется с точностью $2\Delta \leqslant \varepsilon$, и для достижения этой точности требуется $N-2$ вычислений.

Приведем алгоритм поиска на основе чисел Фибоначчи для нахождения экстремума целевой функции:

1. Получить набор или одно альтернативное решение.
2. Ввести границы исследуемого интервала $[a, b]$.
3. По заданной точности рассчитать вспомогательное число N: $N = (b-a)/\delta$, где δ — условный коэффициент ($\delta \geqslant 1$).

4. Найти такое число Фибоначчи F_n, для которого выполняется неравенство $F_{n-1} < N \leqslant F_n$.
5. Определить минимальный шаг поиска $m = \left[\dfrac{b-a}{F_n}\right]$, где $[x]$ — ближайшее меньшее целое x.
6. Выбрать первую точку на отрезке и рассчитать значение целевой функции $f(a)$.
7. Перейти к следующей точке $x_1 = a_n + mF_{n-2}$, в которой рассчитать $f(x_1)$.
8. Если для новой точки $f(x_1) > f(a)$ (шаг удачный), то следующую точку определить как $x_2 = x_1 + mF_{n-3}$; при неудачном шаге ($f(x_1) < f(a)$) определить $x_2 = x_1 - mF_{n-3}$.
9. Последующие шаги выполнить с уменьшающейся величиной шага: $\Delta x_i = \pm mF_{n-i-2}$ для i-го шага.
10. Конец работы алгоритма.

Если при выполнении шага значение целевой функции в точке $x_{i+1} = x_i + \Delta x_i$ лучше, то следующий $(i+1)$ шаг выполняется из точки x_{i+1}: $x_{i+2} = x_{i+1} + \Delta x_{i+1}$. Если i шаг неудачный, т. е. $f(x_{i+1}) < f(x_i)$, то следующий $(i+1)$ шаг выполняется из точки x_i: $x_{i+2} = x_{i+1} - \Delta x_{i+1}$. Процесс продолжается, пока не рассмотрены все числа Фибоначчи в отрезке $[a, b]$ в убывающей последовательности или по заданию ЛПР.

4. Рассмотрим поиск на основе ***метода золотого сечения***.

Этот метод применяется, когда необходимо найти не собственно экстремум, а то значение аргумента функции, при котором выполняется заданное условие, например, $f(x) \geqslant y$. Оценить заранее требуемое число измерений в такой постановке задачи невозможно, поэтому нельзя определить и положение первого эксперимента для поиска по методу Фибоначчи. Метод золотого сечения позволяет избавиться от зависимости первого опыта и от числа опытов. Здесь также сохраняется условие симметричности последовательных экспериментов, как и в методе Фибоначчи, т. е. справедливо соотношение (2.4). Здесь предлагается условие

$$\frac{d_{j-1}}{d_j} = \frac{d_j}{d_{j+1}} = c = \text{const} = \frac{1+\sqrt{5}}{2} \approx 1{,}618,$$

которое означает постоянство отношения длин последовательных отрезков.

При использовании метода золотого сечения первое измерение делается на расстоянии $d_2 = d_1/c$ от любого края интервала неопределенности ($d_1 = b - a$). Второе измерение делается симметричным первому. Поэтому места расположения точек поиска определяют по формулам:

$$x_1 = a + \left[\frac{b-a}{c}\right], \quad x_2 = h - \left[\frac{b-a}{c}\right].$$

Преимущество этого метода состоит в том, что при его использовании каждый новый эксперимент приводит к сокращению интервала неопределенности.

5. Рассмотрим ряд ***численных методов*** решения инженерных задач. Сущность всех численных методов оптимизации состоит в построении последовательности векторов $\{x_k\}$, $k = 0, 1, 2, \ldots$, удовлетворяющих условию $f(x_{k+1}) > f(x_k)$ (в случае поиска максимума) или $f(x_{k+1}) < f(x_k)$ (в случае минимума). К ним относят методы, требующие для своей реализации вычисления первых производных целевой функции $f(x)$. Известно, что градиент скалярной функции $f(x)$ в некоторой точке x_i направлен в сторону наискорейшего возрастания функции. Вектор, противоположный градиенту, $\nabla f(x_i)$ — антиградиент, направлен в сторону наискорейшего убывания функции $f(x)$. Наиболее применимы на практике следующие методы:

- релаксации;
- градиента;
- наискорейшего спуска (подъема).

5.1. ***Метод релаксации*** наиболее прост. Он заключается в определении направления, вдоль которого значение целевой функции изменяется наилучшим образом. Для этого в начальной точке поиска определяются производные оптимизируемой функции $f(x)$ по всем направлениям. Затем выбирается наибольшая по модулю производная и соответствующая ей переменная изменяется до достижения локального оптимума. В новой точке определяются производные по всем остальным переменным, и производится поиск нового локального оптимума и т. д. Скорость движения к локальному оптимуму зависит от выбора величины шага изменения независимых переменных. При «малом» шаге процесс сходится медленно, а при «большой» величине шага возможно блуждание, т. е. проскакивание оптимума. В теории поиск заканчивается, когда все частные производные целевой функции равны нулю. На практике в качестве критерия окончания поиска берется условие:

$$\sum_{j=1}^{m} \left(\frac{\partial f(x)}{\partial x_j} \right)^2 \leqslant \delta,$$

где δ — заданное заранее число.

5.2. В ***градиентном методе*** движение при поиске точки оптимума осуществляется в направлении наибыстрейшего возрастания целевой функции, т. е. в направлении градиента $\operatorname{grad} F(x) = \nabla f(x)$. Основное уравнение градиентного поиска оптимума имеет вид

$$x_{k+1} = x_k + h_k \operatorname{grad}(x_k),$$

где h — шаг поиска.

5.3. Одним из вариантов градиентного метода является ***равномерный поиск*** при $h_k = h_0 = \text{const}$. Он гарантирует приближение к экстремуму на расстояние не более h_0, хотя затем вызывает блуждания в окрестности экстремума. Существует еще пропорциональный градиентный поиск, когда его начинают с определенного значения α. Применив в качестве начальной точки произвольное или заданное значение $\alpha_0 \leqslant 1$, на каждом шаге его уменьшают вдвое, пока не получится $f(x_{k+1}) < f(x_k)$ (при поиске минимума).

Процесс можно остановить, как только приращение $\Delta_{k+1} \leqslant \Delta_0$ сравняется с заранее заданным значением. Отметим, что при использовании градиентного метода на каждом шаге меняется значение не одной, а всех независимых переменных. В каждой точке градиент вычисляется заново, а каждый шаг выполняется в направлении наискорейшего возрастания функции в соответствующей точке.

5.4. В ***методе наискорейшего подъема*** объединены основные идеи методов релаксации и градиента. После нахождения градиента анализируемой функции в начальной точке осуществляется продвижение, пока функция $f(x)$ не перестанет возрастать. Итак, получаем

$$x_{k+1} = x_k + \alpha_k \nabla f(x_k),$$

где α_k — частный максимум функции одной переменной $f[x(\alpha)] = f(\alpha)$. При использовании метода наискорейшего подъема объем вычислений уменьшается и поиск ускоряется.

6. Для решения инженерных задач с большим числом переменных используют ***статистические методы оптимизации***. Основным их отличием от детерминированных методов является введение элемента случайности в процесс поиска экстремума. Для характеристики статистических методов оптимизации используются понятия накопления, адаптации, самообучения. Накопление — это выборка информации на основе пробных шагов о поведении значения целевой функции вблизи рассматриваемой точки для выбора направления поиска. Самообучение — управляемый процесс изменения вероятностных свойств поиска. Выделяют следующие основные статистические методы оптимизации: простые, с накоплением, с адаптацией, с самообучением.

6.1. В ***статистических методах оптимизации с накоплением*** с помощью пробных шагов собирается информация о поведении целевой функции в некоторой окрестности точки x_k. Затем находится направление для рабочего шага, близкого к антиградиентному, причем степень близости зависит в основном от числа пробных попыток. Основной метод в этом классе — метод наилучшей пробы. Из точки x делается q случайных независимых попыток с заданным шагом. Для каждой пробы вычисляется значение целевой функции и шаг совершается в направлении той пробы, что привела к наилучшему значению целевой функции. Очевидно, что при $q \to \infty$ мы найдем оптимальное значение целевой функции. Важным отличием этого метода является

возможность работы при $q < n$, где n — число переменных оптимизируемой функции.

6.2. В ***статистических методах оптимизации с адаптацией*** параметры поиска изменяются в процессе оптимизации. Введение автоматически изменяющегося масштаба поиска улучшает сходимость и точность метода. Масштаб поиска изменяется, чтобы вдали от экстремума шаг поиска увеличивался, а при приближении к нему уменьшался. Автоматическая настройка масштаба шага часто реализуется на использовании результатов последней итерации. Увеличение масштаба шага осуществляется умножением его на коэффициент роста d_1, а уменьшение — делением на d_2, причем $d_1 \geqslant d_2 \geqslant 1$. Основное условие, которое необходимо соблюдать во всех случаях, это избыточность масштаба. При большом масштабе поиск становится менее чувствительным к погрешностям вычислений.

6.3. В ***статистических методах оптимизации с самообучением*** процесс случайного поиска состоит в перестройке вероятностных характеристик случайного вектора для увеличения числа эффективных итераций и уменьшения неэффективных. В таких алгоритмах случайный вектор ξ перестает быть равновероятным, а в процессе поиска получает определенное преимущество в направлении наилучшего шага. Если избранное направление не приводит к успеху, алгоритм с самообучением ищет другое. На начальных итерациях поиск эффективного направления начинается в равновероятной зоне, а затем с набором информации о характере целевой функции последовательно приобретает преимущество в выборе наилучшего направления.

6.4. Опишем кратко основной ***метод случайного поиска — метод Монте-Карло***. При этом необходимо найти экстремум $V(F)$ в заданной области допустимых параметров $V_{\min} \leqslant V \leqslant V_{\max}$ с точностью интервала неопределенности ε. Пусть допустимая область по одному из параметров $D = V_{\max} - V_{\min}$. В методе Монте-Карло вырабатывается псевдослучайный вектор параметров с равномерным распределением. Тогда вероятность непопадания в область экстремума за один шаг равна $P_1(MMK) = 1 - \varepsilon/D$. Соответственно за n шагов алгоритма

$$P_n(MMK) = (1 - \varepsilon/D)^n,$$

а вероятность попадания в область минимума

$$P_n(MMK) = 1 - (1 - \varepsilon/D)^n.$$

Следовательно, число G генераций случайного вектора, необходимых для уточнения минимума с точностью ε, с заданной вероятностью запишется

$$G = \lg(1 - P_n(MMK))/\lg(1 - \varepsilon/D).$$

В случае k внутренних параметров целевой функции число генераций этого вектора увеличится в $\sqrt{k}$ раз и составит $n\sqrt{k}$. А общее

число обращений к модели метода Монте-Карло за G генераций составит

$$n_{\text{ср}} = n\sqrt{k}\,G$$

и определит условие применимости метода при одинаковом числе итераций. Метод Монте-Карло эффективен при большом числе испытаний, $G \geqslant 1000$. При повышенной точности ε число испытаний резко возрастает и метод Монте-Карло неэкономичен. В практических инженерных задачах необходимо комбинировать статистические и детерминированные методы поиска.

7. Рассмотрим ***методы поиска в глубину***, ***в ширину***, ***с возвратом***, ***горизонта***. Эти методы основаны на анализе движений по ***дереву*** решений, пока это возможно. Отметим, что эти методы могут выполняться не только на основе случайного поиска, но и с использованием заданных эвристик.

7.1. ***Метод поиска с возвратом*** может выполняться в двух направлениях, определенных случайным вектором и противоположным ему с заданным шагом b. Тогда значения целевой функции определяются в точках $x_{1,2} = x_0 \pm b$ и в сторону уменьшения осуществляется рабочий шаг на величину h. После удачного шага можно продолжать поиск в том же направлении, а после неудачного — обращаться к случайному вектору, т. е. «штрафовать случайность». В этом методе из-за введения случайности движение к экстремуму производится не только вдоль координат, но и вдоль любого направления. При неудачном шаге в пространстве параметров выбирается точка x_0 и задается постоянный шаг h. Из этой точки в случайном направлении, определенном случайным единичным вектором $\xi = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$, осуществляется шаг h и вычисляется значение целевой функции в новой точке x_1. Если при поиске минимума значение целевой функции в x_1 уменьшилось, то новый шаг из x_1 осуществляется в случайном направлении. При неудачном шаге (возрастании целевой функции) происходит возврат в начальную точку и осуществление нового пробного шага в случайном направлении.

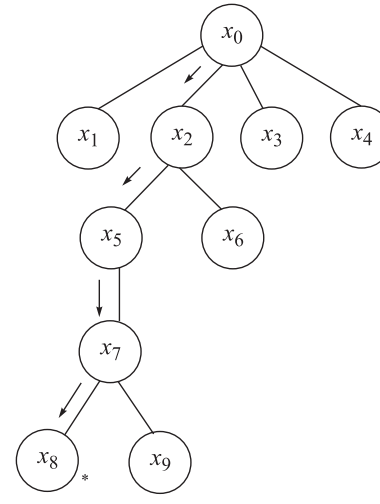


Рис. 2.2. Схема метода поиска в глубину

7.2. ***Метод поиска в глубину*** является, по мнению авторов, частным случаем метода поиска с возвратом. Идея поиска в глубину состоит в том, чтобы в каждой исследуемой вершине дерева решений

выбирать один из возможных путей и исследовать его до конца. Другие пути при этом не рассматриваются, пока сохраняется возможность получить локальный оптимум, исследуя выбранное направление. Пример схемы поиска в глубину показан на рис. 2.2. Здесь знак (*) означает, что задача поиска решена; $x_0, \ldots, x_9$ — исследуемые вершины дерева решений.

Основной недостаток метода поиска в глубину заключается в том, что при исследовании дерева решений с большой вероятностью можно пройти мимо той ветви, на которой раньше всего появляется локальный оптимум. Временна́я сложность алгоритма метода поиска в глубину составляет $\approx O(n) \div O(n^2)$.

7.3. При использовании ***метода поиска в ширину*** ветвление происходит от уровня к уровню. Если на первом уровне начальная задача разбивается на подзадачи, то каждая из них исследуется раньше, чем задачи 2-го уровня. Задачи первого уровня, которые трудны для разрешения, разбиваются на подзадачи уровня 2, и процесс продолжается аналогично. Примерный вид дерева решений при методе поиска в ширину показан на рис. 2.3. Временна́я сложность алгоритма метода поиска в ширину составляет $\approx O(n^2) \div O(n!)$.

7.4. Рассмотрим ***метод горизонта***. Это метод поиска квазиоптимальных решений, заключающийся в ограничении глубины просмотра в каждой точке дерева решений. После достижения промежуточной цели оцениваются дальнейшие возможности, и продолжается процесс поиска. Рассмотрим на примере основную идею реализации метода. Пусть в блоках разбиения расположено несколько вершин графа и требуется разместить еще заданное подмножество вершин. На рис. 2.4 показан граф $G = (X, U)$, $|X| = 9$, который необходимо разбить на три части, причем частичное разбиение уже выполнено. Если для помещения в блок выбирается претендент, наиболее связанный с ранее размещенными элементами, то нет смысла пробовать располагать его во все свободные блоки. Имеет смысл определять для размещения «перспективные» блоки и анализировать процесс расположения в них вершин графа. Соответственно определение перспективных блоков зависит от целевой функции. При минимизации суммарного количества

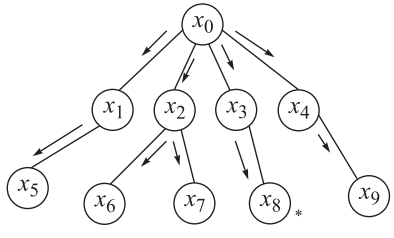


Рис. 2.3. Схема метода поиска в ширину

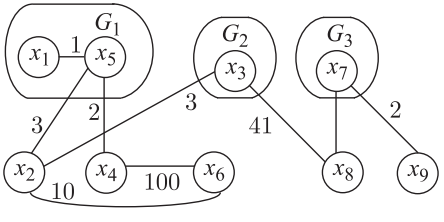


Рис. 2.4. Частичное разбиение графа G на три части

внешних связей можно, например, ограничить «горизонт видимости» путями длины два, три или четыре с ранее установленными вершинами.

Например, на рис. 2.4 вершины x_1, x_5 размещены в G_1, x_3 — в G_2, x_7 — в G_3. В первом блоке должно быть 4 вершины, во втором — 2, в третьем — 3. Горизонт здесь будет состоять из путей длины 2. Работу алгоритма начинаем с анализа вершин x_2, x_4, которые связаны с x_5. Если горизонт состоит из пути длины 1, то более предпочтительна для помещения в блок вершина x_2. При увеличении горизонта до двух в блок G_1 помещаем x_4 и x_6. Продолжая аналогично, получим, что в G_2 попала вершина x_8, а в $G_3 - x_2$, x_9, при этом целевая функция имеет значение $K = 18$.

Окончательное разбиение графа G на три части показано на рис. 2.5. Временна́я сложность алгоритма лежит в пределах $O(n) - O(n^4)$. Как видно, временна́я сложность алгоритма и точность получения результата находятся как бы в противоречии, так как увеличение горизонта просмотра приводит к более качественным результатам за большее время.

7.5. При решении инженерных задач большой размерности часто применяется разбиение задачи на составляющие части. В этом случае выделяют три способа ветвления дерева решений: И; ИЛИ; комбинация И, ИЛИ. Пусть у нас имеется задача Q, подлежащая разбиению на подзадачи Q_1, Q_2, Q_3 (рис. 2.6). Если для решения Q необходимо решить все подзадачи Q_1, Q_2, Q_3, Q_4, Q_5, то это ветвление И. Если решение Q_3 возможно через решения Q_6 или Q_7, то это ветвление ИЛИ. Примером ветвления (поиска) И является одновременное решение всех подзадач. Примером поиска ИЛИ является решение одной или другой подзадачи.

7.6. Для получения точных решений практических задач небольшой размерности ($n < 100$) эффективно применяется ***метод ветвей и границ***. Он заключается в том, что множество всех возможных решений разбивается на подмножества, последовательно уменьшающиеся

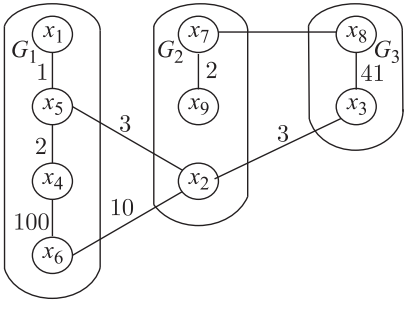


Рис. 2.5. Окончательное разбиение графа G

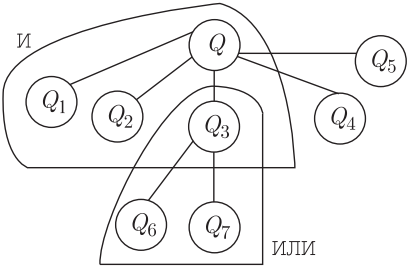


Рис. 2.6. Пример схемы поиска на основе И–ИЛИ деревьев

по числу заключенных в них решений. В процессе разбиения вычисляется нижняя (верхняя) оценка, характеризующая данное подмножество. Нижняя оценка вычисляется, если ищется глобальный минимум, а верхняя — если максимум. В результате получаем подмножество, содержащее единственное решение, являющееся оптимальным. Например, необходимо решить оптимизационную задачу поиска глобального минимума. Известно, что Z — множество решений данной задачи. Каждое решение z из данного множества оценивается с помощью целевой функции $f(z)$. Требуется найти такое решение $t \in Z$, для которого $f(t)$ — наименьшее среди всех $f(z)$, $z \in Z$. Использование специфики конкретной задачи позволяет в ряде случаев найти нижнюю границу значения целевой функции f на множестве решений Z или на каком-либо его подмножестве $z' \subseteq Z$. Нижней границей $\theta(z)$ или $\theta(z')$ целевой функции f будем называть значение, которое меньше или равно значению целевой функции $f(z)$ для любого решения $z \in Z$ или $z \in Z'$. Таким образом, $f(z) \geqslant \theta(z)$ для любого $z \in Z$ или $f(z) \geqslant \theta(z')$ для любого $z \in Z'$.

Решение задачи заключается в разбиении всего множества решений Z на подмножества $Z_1, Z_2, \ldots, Z_s$, для которых

$$Z_1 \cap Z_2 \cap \ldots \cap Z_s = \varnothing, \quad Z_1 \cup Z_2 \cup \ldots \cup Z_s = Z.$$

Так как $Z_i \subseteq Z$, то минимальное значение целевой функции $f(t)$ решения t из подмножества Z_i множества решений Z больше или равно минимальному значению целевой функции $f(t)$, если $t \in Z$. Нижняя граница значения целевой функции для подмножества Z_i не меньше нижней границы для множества решений Z, т. е. $\theta(z_i) \geqslant \theta(z)$, $i = 1, 2, \ldots, s$. Метод ветвей и границ эффективен, если при разбиении множества решений на подмножества Z_i удается улучшать оценки, т. е. получить строгое неравенство $\theta(z_i) > \theta(z)$.

Разбиение множества решений на подмножества представляется с помощью дерева решений на основе матрицы, ***графа*** или списка. В качестве исходной вершины дерева решений принимается множество Z. Вычисляем нижнюю границу $\theta(z)$ множества Z. Если найдем такое решение t, что $f(t) = \theta(z)$, то t — искомое оптимальное решение. Если найти его не удалось, то множество Z разбиваем на ряд подмножеств, образующих на первом шаге вершины дерева, которые обозначим $Z_1^1, Z_2^1, \ldots, Z_s^1$. Верхний индекс при Z соответствует номеру шага, нижний определяет номер подмножества множества решений на данном шаге. Эту процедуру назовем *ветвлением*. Определяем нижние границы $\theta(Z_j^1)$, $j = 1, 2, \ldots, s$. Если найдено такое решение $t \in Z_p'$, что $f(t) = \theta(Z_p^l)$, причем $\theta(Z_p^l) \leqslant \theta(Z_j^l)$, $j \neq p$; $j = 1, 2, \ldots, s_1$, то t — искомое оптимальное решение. Если найти его не удалось, то выбираем из подмножеств $Z_1^1, Z_2^1, \ldots, Z_{s1}^1$ подмножество Z_i^1 по правилу $\theta(Z_i^1) = \min \theta(Z_j^1)$, $j = 1, 2, \ldots, s_2$, и разбиваем его на определенное

число подмножеств:

$$Z_i^1 = Z_{i1}^1 \cup Z_{i2}^1 \cup \ldots \cup Z_{in}^1.$$

Все подмножества $Z_1^1, Z_2^1, \ldots, Z_{i-1}^1, Z_{i+1}^1, \ldots, Z_{s1}^1, Z_{i1}^1, Z_{i2}^1 \cup \ldots \cup Z_{in}^1$, ранее не подвергавшиеся разбиению, обозначаем $Z_1^2, Z_2^2, \ldots, Z_{s2}^2$.

Переходим ко второму шагу и определяем нижние границы $\theta(Z_j^2)$, $j = 1, 2, \ldots, s_2$. Если найдено такое решение $t \in Z_p^2$, что $f(t) = \theta(Z_p^2)$, причем $\theta(Z_p^2) \leqslant \theta(Z_j^2)$, $j \neq p$; $j = 1, 2, \ldots, s_2$, то t — искомое решение. Если найти его не удалось, то выбираем из подмножеств $Z_1^2, Z_2^2, \ldots, Z_{S2}^2$, подмножество Z_i^2 по правилу $\theta(Z_i^2) = \min \theta(Z_j^2)$, $j = 1, 2, \ldots, s_2$, и разбиваем Z_i^2 на определенное число подмножеств:

$$Z_i^2 = Z_{i1}^2 \cup Z_{i2}^2 \cup \ldots \cup Z_{in}^2.$$

Все подмножества $Z_1^2, Z_2^2, \ldots, Z_{i-1}^2, Z_{i+1}^2, \ldots, Z_{s1}^2, Z_{i1}^2, Z_{i2}^2 \cup \ldots \cup Z_{in}^2$, ранее не подвергавшиеся разбиению, обозначаем: $Z_1^3, Z_2^3, \ldots, Z_{s3}^3$ — и переходим к 3-му шагу и т. д. Если среди подмножеств решений v-го шага удается найти решение $t \in Z_p^v$, такое, что

$$f(t) = \theta(Z_p^v) \leqslant \theta(Z_j^v), \quad j \neq p; \quad j = 1, 2, \ldots, s_v,$$

то t — искомое оптимальное решение. В противном случае по правилу

$$\theta(Z_j^v) = \min \theta(Z_j^v), \quad j = 1, 2, \ldots, s_v,$$

определяем $\theta(Z_j^v)$ и разбиваем его на ряд подмножеств. Обозначаем еще не подвергавшиеся разбиению множества как подмножества $v + 1$-го шага, и продолжаем процесс. Так как множество Z конечно, то в результате получим подмножества, состоящие из одного решения. Среди них будет найдено оптимальное решение t.

Метод ветвей и границ позволяет также находить квазиоптимальные решения с заданной точностью Δ. Если на каком-либо шаге μ найдено решение t^1 такое, что $f(t^1) - \theta(Z_r^\mu) \leqslant \Delta$, где Z_r^μ — подмножество с наименьшей нижней границей среди всех подмножеств данного шага, то t^1 — искомое квазиоптимальное решение.

Рассмотрим пример ветвления дерева решений. Пусть на первом шаге множество решений Z разбито, например, на три подмножества Z_1^1, Z_2^1 и Z_3^1. Среди них оптимальное решение t не найдено. Сравнивая оценки каждой вершины дерева решений, определяем, что дальнейший поиск осуществляется из вершины Z_2^1. Подмножество Z_2^1 разбивается на три подмножества: Z_{21}^1, Z_{22}^1 и Z_{23}^1. Первый шаг разбиения дерева решений показан на рис. 2.7.

Вершины Z_1^1, Z_3^1, Z_{21}^1, Z_{22}^1, Z_{23}^1, которые не рассматривались, обозначаем соответственно Z_1^2, Z_2^2, Z_3^2, Z_4^2, Z_5^2. Находим нижние границы для этих подмножеств. Если оптимальное решение t не найдено, то вершина Z_1^2: $Z_1^2 = Z_{11}^2 \cup Z_{12}^2$ подвергается ветвлению. Дерево решений, соответствующее второму шагу алгоритма, показано на рис. 2.8.

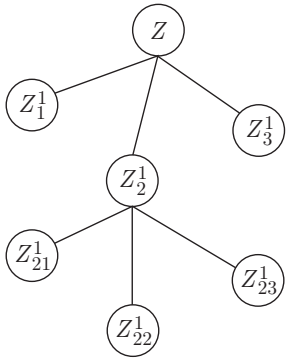


Рис. 2.7. Первый шаг построения дерева решений

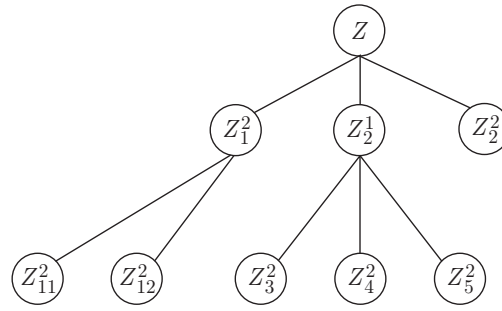


Рис. 2.8. Второй шаг построения дерева решений

Обозначаем вершины, находим границы для подмножеств третьего шага. Если оптимальное решение t не найдено, ветвится, например, вершина Z_6^3 (рис. 2.9).

Процесс повторяется до получения оптимального или квазиоптимального решения. Таким образом, для эффективного применения метода ветвей и границ необходимо, учитывая специфику решения, получить выражение для вычисления нижних границ целевой функции. Это даст число ветвящихся вершин и, следовательно, степень сокращения перебора при решении задачи.

7.7. Одной из разновидностей точных и приближенных к ним методов нахождения решений является ***метод моделирования отжига***.

В 1953 г. Метрополис предложил вычислительную процедуру, воспроизводящую механизм отжига металлов, для моделирования состояния равновесия сложных систем при заданной конечной «температуре».

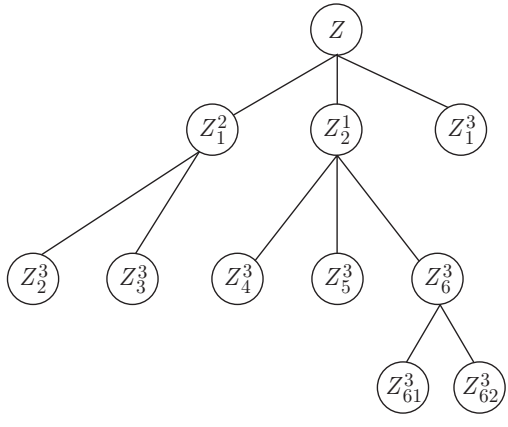


Рис. 2.9. Третий шаг построения дерева решений

Идея переноса механизма отжига металлов на решение оптимизационной задачи состоит в том, что процесс оптимизации связывают с некоторой температурой. На каждом шаге поиска случайным образом осуществляется малое изменение состояния объекта и вычисляется изменение ΔE энергии системы. Новая конфигурация системы принимается с вероятностью 1, если $\Delta E \leqslant 0$, и с вероятностью, равной $\exp(-\Delta E/kt)$, если $\Delta E > 0$. Эта процедура переносится на решение оптимизационных задач. При этом состояния физической системы заменяются изменением критерия качества, а значение kt заменяется обобщенным понятием «температура» T, которая может рассматриваться как управляющий параметр оптимизационной процедуры.

На начальном этапе температуру принимают высокой, а затем ее ступенчато снижают. При каждой температуре выполняют серию пробных переборов решений, и после каждой перестановки подсчитывается значение целевой функции. Лучшие решения принимаются с вероятностью 1, а «плохие», для которых значение целевой функции ухудшается, принимаются с некоторой вероятностью. Такой вероятностный механизм дает возможность, принимая в качестве исходных некоторые «плохие» решения, проскакивать через локальные оптимумы и находить глобальные.

Схема выполнения алгоритма моделирования отжига показана на рис. 2.10.

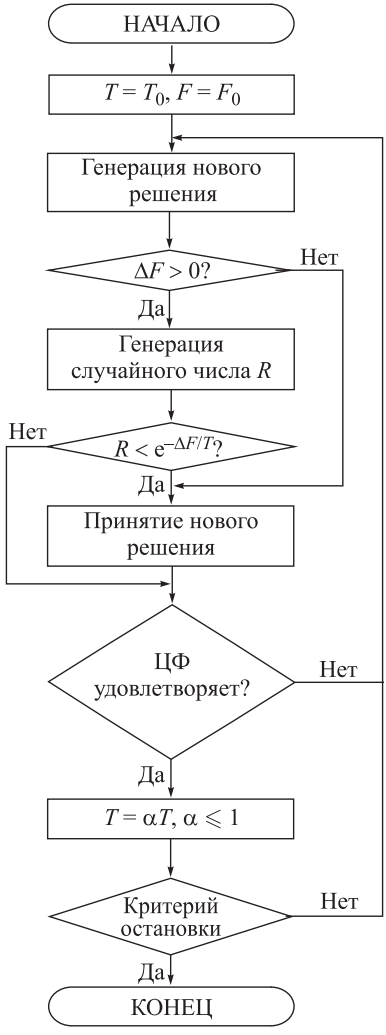


Рис. 2.10. Схема алгоритма моделирования отжига

Реализация вероятностного механизма в оптимизационном алгоритме на основе моделирования отжига осуществляется следующим образом. Если в результате перебора элементов произошло улучшение значения целевой функции F, то генерируется случайное равномерно распределенное число R в диапазоне $[0, 1]$ и проверяется условие

$$R < \exp(-\Delta F/T),$$

где ΔF — приращение целевой функции; T — температура на заданном шаге работы алгоритма. Если условие выполняется, то полученное решение принимается за новое исходное, в противном случае исходным остается старое решение.

Процедура моделирования отжига требует больших вычислительных затрат. Для повышения скорости можно выбрать одну ветвь дерева решений и провести моделирование отжига, как усложненный поиск в глубину.

7.8. Опишем алгоритм общей нечеткой стратегии решения оптимизационных задач на основе ***комбинированного поиска***.

1. Выбрать начальное приближение (решение инженерной задачи).
2. Определить динамику изменения значений целевой функции в зависимости от времени или числа элементов математической модели.
3. Определить вектор направления движения.
4. Выполнить оптимальный шаг по выбранному направлению — одномерный поиск.
5. Если целевая функция $\to \min$ (при минимизации), то перейти к 3. Иначе перейти к 6.
6. Если выходные параметры задачи удовлетворяют заданным требованиям, то окончить оптимизацию. В противном случае изменить значение целевой функции, весовые коэффициенты, параметры, ввести несколько уровней адаптации и перейти к 3.
7. Конец работы алгоритма.

2.3. Выводы

В соответствии с вышесказанным можно условно выделить три типа ***поисковых методов***:

1. Вычислительные. Они подразделяются на четыре основных класса:
   - непрямые методы: используются для нахождения локальных экстремумов. При этом решается множество линейных или нелинейных уравнений. Данные методы находят экстремум функции, анализируя ограниченное пространство точек во всех направлениях;
   - прямые методы: осуществляют поиск оптимума путем градиентных алгоритмов. Даже для двух локальных оптимумов это большая проблема — найти лучшее решение, а для многоэкстремальных задач применение этих методов затруднительно. Сейчас применяются методы линейного, нелинейного и динамического программирования;
   - точные методы;
   - приближенные методы.
2. Невычислительные.

3. Случайные.

Для определения глобального минимума значений целевой функции в многоэкстремальных задачах предлагается:

- учитывать знания о решаемых задачах, позволяющие связать возможности значения целевой функции с известными значениями в точках реализованных испытаний;
- изменять начальную точку спуска методом проб и ошибок, градиентными методами, совместными методами локального поиска;

2.4. Контрольные вопросы

1. Что такое «оптимальность»?
2. Опишите понятие «оптимизационная задача».
3. Определите понятия «моделирование» и «модель».
4. Опишите правила построения математической модели исследуемого объекта.
5. Каким образом осуществляется создание математической модели исследуемого объекта?
6. Опишите требования, предъявляемые к математическим моделям.
7. Что такое «адекватность» и «верификация математической модели исследуемого объекта»?
8. Определите понятие точности математической модели.
9. Как определяется погрешность математической модели?
10. Как определить степень универсальности математической модели?
11. Какими параметрами характеризуется экономичность математической модели?
12. Как определяются условия и область работоспособности решаемой задачи?
13. Приведите классификацию математических моделей.
14. Что такое «критерий оптимизации»?
15. Что является целью оптимизационной задачи?
16. Каким образом строится целевая функция?
17. Дайте понятие критерия оптимальности.
18. Дайте понятие экстремума и оптимума целевой функции.
19. Что такое «локальный и глобальный оптимум»?
20. Поясните понятие пространства решений оптимизационной задачи.
21. Каким образом определяется квазиоптимальное решение оптимизационной задачи?
22. Приведите классификацию стратегий поиска.
23. В чем заключается одномерный поиск решений оптимизационной задачи?
24. Опишите метод дихотомии.

25. В чем состоит идея поиска решений оптимизационной задачи на основе метода Фибоначчи?
26. Каким образом выполняется поиск решений на основе метода золотого сечения?
27. Приведите численные методы решения практических задач.
28. В чем заключается метод релаксации?
29. Опишите градиентный метод, равномерный поиск и метод наискорейшего подъема.
30. Приведите основную идею статистических методов оптимизации.
31. Опишите метод Монте-Карло.
32. В чем заключается метод поиска с возвратом?
33. Приведите основную идею метода поиска в глубину.
34. Приведите основную идею метода поиска в ширину.
35. Опишите метод горизонта для поиска решений оптимизационной задачи.
36. Приведите основные способы поиска на основе И–ИЛИ деревьев.
37. Опишите основную идею метода ветвей и границ.
38. В чем заключается метод моделирования отжига?

2.5. Упражнения

1. Постройте математическую модель для решения задачи нахождения максимума функции $y = ax^2 + b$, где $0 < a, b < \infty$, $ab = 2000$.

2. Постройте математическую модель для решения задачи нахождения минимума функции $y = a(x^2 + 2x + b)$, где $0 < a, b < \infty$, $ab = 1000$.

3. Постройте математическую модель для нахождения максимума тестовой функции Де Йонга $f(x_i) = \sum_{1}^{3} x_i^2$ при ограничениях $-5{,}12 \leqslant x_i \leqslant 5{,}12$.

4. Постройте математическую модель для нахождения минимума тестовой функции Де Йонга $f(x_i) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$ при ограничениях $-2{,}048 \leqslant x_i \leqslant 2{,}048$.

5. Постройте математическую модель для нахождения локальных максимумов тестовой функции Де Йонга

$$f(x_i) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

при ограничениях $-2{,}048 \leqslant x_i \leqslant 2{,}048$.

6. Постройте математическую модель для нахождения локальных максимумов тестовой функции Де Йонга $f(x_i) = \sum_{1}^{5} \text{integer}\ (x_i)$, при ограничениях $-5{,}12 \leqslant x_i \leqslant 5{,}12$.

7. Постройте математическую модель для нахождения локальных минимумов тестовой функции Де Йонга $f(x_i) = \sum_{1}^{5} \text{integer}\ (x_i)$ при ограничениях $-5{,}12 \leqslant x_i \leqslant 5{,}12$.

8. Постройте математическую модель для нахождения локальных максимумов тестовой функции Де Йонга

$$f(x_i) = 0{,}002 + \sum_{j=1}^{25} \frac{1}{j + \sum\limits_{i=1}^{2} (x_i - a_{ij})^6}$$

при ограничениях $-65{,}536 \leqslant x_i \leqslant 65{,}536$.

9. Определите адекватность построенной математической модели из упр. 3.

10. Определите точность построенной математической модели из упр. 4.

11. Определите экономичность построенной математической модели из упр. 5.

12. Определите область работоспособности для упр. 3.

13. Постройте нечеткий алгоритм математического описания следующей задачи.

Дано: 5 космонавтов и таблица их психологической совместимости (оценка проводилась по 5-бальной шкале)

0	4	5	3	1
4	0	3	5	2
5	3	0	2	4
3	5	2	0	3
1	2	4	3	0

Определите пару космонавтов, обладающих наибольшей психологической совместимостью.

14. Найдите квазиоптимальное решение для функции из упр. 8.

15. Найдите максимум тестовой функции из упр. 3 на основе последовательного поиска.

16. Найдите минимум тестовой функции из упр. 4 на основе метода дихотомии.

17. Найдите минимум тестовой функции из упр. 6 на основе метода Фибоначчи.

18. Постройте алгоритм нахождения оптимальных значений функции на основе метода «золотого сечения».

19. Найдите минимум тестовой функции из упр. 6 на основе метода «золотого сечения».

20. Постройте алгоритм нахождения оптимальных значений функции на основе метода релаксации.

21. Найдите локальный минимум тестовой функции из упр. 7 на основе метода релаксации.

22. Постройте алгоритм нахождения оптимальных значений функции на основе градиентного метода.

23. Найдите локальный минимум тестовой функции из упр. 6 на основе градиентного метода.

24. Постройте алгоритм нахождения оптимальных значений функции на основе метода наискорейшего подъема.

25. Найдите локальный максимум тестовой функции из упр. 7 на основе метода наискорейшего подъема.

26. Постройте алгоритм нахождения оптимальных значений функции на основе метода случайного поиска (метод Монте-Карло).

27. Найдите локальный максимум тестовой функции из упр. 8 на основе метода случайного поиска (метод Монте-Карло).

28. Постройте алгоритм нахождения оптимальных значений инженерных задач на основе поиска в глубину.

29. Постройте алгоритм нахождения оптимальных значений графовых задач на основе поиска в ширину.

30. Приведите алгоритм нахождения оптимальных значений инженерных задач на основе поиска с возвратом.

31. Постройте алгоритм нахождения оптимальных значений графовых задач на основе метода горизонта.

32. Определите кратчайший путь из вершины 1 графа в вершину 7 на основе поиска в глубину. Матрица длин путей задана:

0	5	7	3	8	4	2	1
5	0	3	9	5	10	12	2
7	3	0	7	11	2	15	3
3	9	7	0	13	6	4	4
8	5	11	13	0	7	9	5
4	10	2	6	7	0	8	6
2	12	15	4	9	8	0	7
1	2	3	4	5	6	7	

33. Определите кратчайший путь из вершины 1 графа в вершину 7 на основе поиска в ширину. Матрица длин путей задана:

0	5	7	11	8	4	5
5	0	3	3	5	10	11
7	3	0	7	11	4	15
11	3	7	0	13	8	4
8	5	11	13	0	7	14
4	10	4	8	7	0	2
5	11	15	4	14	2	0

34. Определите кратчайший путь из вершины 1 графа в вершину 7 на основе поиска с возвратом. Матрица длин путей задана:

0	9	7	11	8	10	5
9	0	8	3	5	3	11
7	8	0	5	11	4	7
11	3	5	0	4	8	4
8	5	11	4	0	7	14
10	3	4	8	7	0	2
5	11	7	4	14	2	0

35. Определите кратчайший путь из вершины 1 графа в вершину 7 на основе метода горизонта. Матрица длин путей задана:

0	3	7	9	8	5	7
3	0	8	3	5	6	11
7	8	0	5	11	12	7
9	3	5	0	4	8	10
8	5	11	4	0	7	8
5	6	12	8	7	0	2
7	11	7	10	8	2	0

36. Постройте алгоритм нахождения оптимальных значений инженерных задач на основе метода ветвей и границ.

37. Найдите оптимальное решение задачи

$$F = 6x_1 + 4x_2 \to \max,$$
$$7x_1 + 2x_2 \leqslant 16,$$
$$5x_1 + 8x_2 \leqslant 32,$$
$$x_1, x_2 \text{ — целые}$$

на основе метода ветвей и границ.

38. Постройте алгоритм нахождения оптимальных значений графовых задач на основе метода моделирования отжига.

39. Найдите оптимальное решение задачи

$$F = 6x_1 + 4x_2 \to \max,$$
$$7x_1 + 2x_2 \leqslant 16,$$
$$5x_1 + 8x_2 \leqslant 32,$$
$$x_1, x_2 \text{ — целые}$$

на основе метода моделирования отжига.

Глоссарий к разделу 2

Адекватность математической модели — определенное соответствие модели решаемой задаче или процессу принятия решений.

Верификация модели — проверка ее истинности, адекватности; сводится к сопоставлению расчетных результатов по модели с соответствующими данными действительности.

Глобальный максимум (***минимум***) — значение целевой функции, которое больше (меньше) всех остальных.

Граф — математический объект, состоящий из множества вершин и множества ребер, находящихся в заданном отношении.

Дерево — связный граф без циклов.

Квазиоптимальное решение — одно из множества решений, попадающих в локальный экстремум, близкий к глобальному.

Критерий или целевая функция — мера оценки исследуемого явления.

Критерий оптимальности — признак, по которому функционирование системы признается наилучшим из возможных вариантов ее функционирования. Показатель, выражающий предельную меру экономического эффекта принимаемого решения для сравнительной оценки возможных альтернатив и выбора наилучшего из них.

Локальный максимум (минимум) — один из возможных максимумов (минимумов) в пространстве решений.

Математическая модель — выражения, отображающие зависимость между входными и выходными переменными, а также перемен-

ными состояния. Система математических соотношений, описывающих изучаемые процессы или явления.

Метод ветвей и границ заключается в том, что множество всех возможных решений разбивается на подмножества, последовательно уменьшающиеся по числу заключенных в них решений. В процессе разбиения вычисляется нижняя (верхняя) оценка, характеризующая данное подмножество. В результате получаем подмножество, содержащее единственное решение, являющееся оптимальным.

Метод горизонта заключается в ограничении глубины просмотра в каждой точке дерева решений; позволяет достигать промежуточной цели, а затем оценивать дальнейшие возможности и продолжать или усовершенствовать процесс поиска.

Метод градиента — движение при поиске точки оптимума осуществляется в направлении наибыстрейшего возрастания целевой функции.

Метод дихотомии — реализуется за счет механизма обычного перебора возможных точек разрыва.

Метод золотого сечения — позволяет избавиться от зависимости первого опыта и от числа опытов. Здесь используется условие, что соотношение длины соседних выборок равно числу 1,618.

Метод моделирования отжига состоит в том, что процесс оптимизации связывают с некоторой температурой. При каждой температуре выполняют серию пробных переборов решений и после каждой перестановки подсчитывается значение целевой функции. Лучшие решения принимаются, а «плохие», для которых значение целевой функции ухудшается, принимаются с некоторой вероятностью.

Метод наискорейшего подъема — комбинация методов релаксации и градиента. После нахождения градиента анализируемой функции в начальной точке осуществляется продвижение, пока целевая функция не перестанет возрастать.

Метод поиска в глубину состоит в том, чтобы в каждой исследуемой точке дерева решений выбирать один из возможных путей и исследовать его до конца.

Метод поиска в ширину — ветвление дерева решений происходит от уровня к уровню, пока не будет получено решение задачи.

Метод поиска с возвратом — выполняется в двух направлениях, определенных случайным вектором. После удачного шага поиск продолжается в том же направлении, а после неудачного осуществляется возврат на предыдущий шаг.

Метод релаксации заключается в определении направления, вдоль которого значение целевой функции изменяется наилучшим образом.

Метод случайного поиска (***метод Монте-Карло***) — выбор псевдослучайного вектора параметров с равномерным распределением.

Метод Фибоначчи состоит в том, что выборки, проводимые последовательно, располагаются так, чтобы для соседних выборок вы-

полнилось то условие, что очередной член ряда равен сумме двух предыдущих, кроме первого и второго.

Моделирование — это исследование явлений, процессов путем построения их моделей. Использование моделей для определения поведения и характеристик реальных событий.

Модель — это копия или аналог изучаемого явления или процесса, отражающая существенные свойства моделируемого объекта с точки зрения исследователя. Приближенное описание какого-либо класса явлений внешнего мира.

Область работоспособности — часть пространства управляемых переменных, в которой выполняются условия работоспособности.

Оптимальность — это оценочное отражение субъективного свойства через некоторое количественное соотношение, т. е. количественное значение качества, которое желательно придать моделируемой задаче.

Оптимизационная задача — задача, в которой необходимо найти решение, в некотором смысле наилучшее или оптимальное. Это кортеж длины три, состоящий из пространства решений, ограничений и области допустимых решений.

Пассивный поиск — случайный выбор пар на заданном отрезке для нахождения оптимума целевой функции.

Погрешность математической моделей — степень отклонения реальной модели от эталонного значения.

Последовательный поиск выполняется путем перебора значений целевой функции для нахождения оптимального значения.

Пространство решений — исходное множество альтернативных вариантов решений.

Результативность — связь между достигнутым результатом и использованными ресурсами.

Статистические методы оптимизации основаны на введении элемента случайности в процесс поиска экстремума.

Степень универсальности математической модели — полнота учета свойств реальной задачи.

Стратегия — оптимальный набор правил и приемов, которые позволяют реализовать общую цель, достигнуть глобальных и локальных целей задачи.

Точность математической модели — степень совпадения значений переменных реального объекта и значений тех же переменных, полученных на основе исследуемой модели.

Условия работоспособности — система соотношений между выходными переменными и их предельно допустимыми по техническому заданию значениями.

Экономичность математической модели характеризует затраты вычислительных ресурсов ЭВМ при реализации модели.

Эксперимент — метод научного познания, предполагающий активное вмешательство в ситуацию, изменяя ее факторы и отмечая изменения в изучаемом объекте.

Экстремум — максимум и минимум значения целевой фукции в оптимизационных задачах.

Эффективность — степень реализации запланированной деятельности и достижения запланированных результатов.

Список литературы к разделу 2

1. *Алексеев О. В. и др.* Автоматизация проектирования радиоэлектронных средств. – М.: Высшая школа, 2000.
2. *Банди Б.* Методы оптимизации. — М.: Радио и связь, 1988.
3. *Батищев Д. И.*, *Львович Я. Е.*, *Фролов В. Н.* Оптимизация в САПР. — Воронеж: Изд-во ВГУ, 1997.
4. *Ботвинник М. М.* О решении неточных переборных задач. — М.: Сов. радио, 1979.
5. *Карелин В. П.*, *Родзин С. И.* УМП по методам математического программирования (поисковой оптимизации). — Таганрог: Изд-во ТРТУ, 1999.
6. *Курицкий Б. Я.* Оптимизация вокруг нас. — Л.: Машиностроение, 1989.
7. *Ларичев О. И.* Теория и методы принятия решений. — М.: Логос, 2000.
8. *Лопатников Л. И.* Экономико-математический словарь. — М.: Наука, 1987.
9. *Нильсон Н.* Принципы искусственного интеллекта. — М.: Радио и связь, 1985.
10. *Растригин Л. А.* Статистические методы поиска. — М.: Наука, 1968.
11. *Goldberg David E.* Genetic Algorithms in Search, Optimization and Machine Learning. — USA: Addison-Wesley Publishing Company, Inc., 1989.

Стремление к оптимизации — это естественное состояние природы. Чтобы задача имела оптимальное решение, необходимо существование множества решений. Поиск оптимального решения основан на построении математической модели, включающей целевую функцию, ограничения и граничные условия, и на ее дальнейшем анализе

3. ОСНОВНЫЕ ПОНЯТИЯ И СТРУКТУРА ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Метод обнаружения истины является аналитическим и сводится к четырем правилам:
1. Принимать за истину лишь то, что мыслится как нельзя более ясно и отчетливо, исключая всякое сомнение в своей истинности;
2. Расчленять каждую проблему на максимально возможное количество частей и разрешать каждую из них наилучшим возможным способом;
3. Вести размышления надлежащим образом, начиная с простейшего и продвигаясь малопомалу, или постепенно, к познанию наиболее сложного;
4. Подводить итоги столь полно и обозревать проблему столь всеобъемлюще, чтобы быть полностью уверенным в том, что не упущено ничего.

Р. Декарт

3.1. Определения и понятия генетических алгоритмов

1. ***Генетические алгоритмы*** — это новая область исследований, которая появилась в результате работ Д. Холланда и его коллег. Генетические алгоритмы, описанные Д. Холландом, заимствуют в своей терминологии многое из естественной генетики. Далее будут приведены технические толкования терминов из биологии и генетики, которые используются в теории и практике генетических алгоритмов. Впервые генетические алгоритмы были применены к таким научным проблемам, как распознавание образов и оптимизация. Генетический алгоритм представляет собой адаптивный поисковый метод, основанный на селекции лучших элементов в популяции, подобно эволюционной теории Ч. Дарвина.

Основой для возникновения генетических алгоритмов послужили модель биологической эволюции и методы случайного поиска. Л. Растригин отмечал, что случайный поиск возник как реализация простейшей модели эволюции, когда случайные мутации моделирова-

лись случайными шагами оптимального решения, а отбор — «устранением» неудачных вариантов.

Эволюционный поиск с точки зрения преобразования информации — это последовательное преобразование одного конечного нечеткого множества промежуточных решений в другое. Само преобразование можно назвать алгоритмом поиска, или генетическим алгоритмом. Генетические алгоритмы — это не просто случайный поиск. Они эффективно используют информацию, накопленную в процессе эволюции.

Цель генетических алгоритмов состоит в том, чтобы:

- абстрактно и формально объяснять адаптацию процессов в естественной системе и интеллектуальной исследовательской системе;
- моделировать естественные эволюционные процессы для эффективного решения оптимизационных задач науки и техники.

В настоящее время используется новая парадигма решений оптимизационных задач на основе генетических алгоритмов и их различных модификаций. Генетические алгоритмы осуществляют поиск баланса между эффективностью и качеством решений за счет «выживания сильнейших альтернативных решений» в неопределенных и нечетких условиях.

Генетические алгоритмы отличаются от других оптимизационных и поисковых процедур следующим:

- работают в основном не с параметрами задачи, а с закодированным множеством параметров;
- осуществляют поиск не путем улучшения одного решения, а путем использования сразу нескольких альтернатив на заданном множестве решений;
- используют ***целевую функцию***, а не ее различные приращения для оценки качества принятия решений;
- применяют не детерминированные, а вероятностные правила анализа оптимизационных задач.

Для работы генетических алгоритмов выбирают множество натуральных параметров оптимизационной проблемы и кодируют их в последовательность конечной длины в некотором алфавите. Они работают до тех пор, пока не будет выполнено заданное число генераций (итераций алгоритма) или на некоторой генерации будет получено решение определенного качества, или когда найден локальный оптимум, т. е. возникла преждевременная сходимость и алгоритм не может найти выход из этого состояния. В отличие от других методов оптимизации эти алгоритмы, как правило, анализируют различные области пространства решений одновременно и поэтому они более приспособлены к нахождению новых областей с лучшими значениями целевой функции.

2. Приведем некоторые понятия и определения из теории генетических алгоритмов. Все генетические алгоритмы работают на основе начальной информации, в качестве которой выступает популяция альтернативных решений P. ***Популяция*** $P^t = \{P_1, P_2, \ldots, P_i, \ldots, P_{Np}\}$

есть множество элементов P_i, $t = 0, 1, 2, \ldots$, — номер генерации генетического алгоритма, N_p — размер популяции. Каждый элемент этой популяции P_i, как правило, представляет собой одну или несколько хромосом или особей, или индивидуальностей (альтернативных упорядоченных или неупорядоченных решений). Хромосомы состоят из генов $P_i = \{g_1, g_2, \ldots, g_v\}$ (элементы, части закодированного решения), и позиции генов в хромосоме называются лоци или локус для одной позиции, т. е. ген — подэлемент (элемент в хромосоме), ***локус*** — позиция в хромосоме, ***аллель*** — функциональное значение гена.

Гены могут иметь числовые или функциональные значения. Обычно эти числовые значения берутся из некоторого алфавита. Генетический материал элементов обычно кодируется на основе двоичного алфавита $\{0, 1\}$, хотя можно использовать буквенные, а также десятичные и другие алфавиты. Примером закодированной хромосомы длины девять на основе двоичного алфавита может служить хромосома $P_i = (001001101)$.

Элементы в генетических алгоритмах часто называют родителями. ***Родители*** выбираются из популяции на основе заданных правил, а затем смешиваются (скрещиваются) для производства «***детей***» (***потомков***). Дети и родители в результате генерации, т. е. одного цикла (подцикла) эволюции, создают новую популяцию. Генерация, то есть процесс реализации одной итерации алгоритма, называется поколением.

По аналогии с процессами, происходящими в живой природе и описанными в разделе 1, в технике считают, что ***эволюция популяции*** — это чередование поколений, в которых хромосомы изменяют свои значения так, чтобы каждое новое поколение наилучшим способом приспосабливалось к внешней среде. Тогда общая генетическая упаковка называется ***генотипом***, а организм формируется посредством связи генетической упаковки с окружающей средой и называется ***фенотипом***.

Каждый элемент в популяции имеет определенный уровень качества, который характеризуется значением целевой функции (в литературе иногда называется функция полезности, приспособленности или пригодности (fitness)). Эта функция используется в генетических алгоритмах для сравнения альтернативных решений между собой и выбора лучших.

Следовательно, основная задача генетических алгоритмов состоит в оптимизации целевой функции. Другими словами, генетические алгоритмы анализируют популяцию хромосом, представляющих комбинацию элементов из некоторого множества, и оптимизируют целевую функцию, оценивая каждую хромосому. Генетические алгоритмы анализируют и преобразовывают популяции хромосом на основе механизма натуральной эволюции. Каждая популяция обладает наследственной изменчивостью. Это означает наличие возможностей случайных отклонений от наиболее вероятного среднего значения целевой функции.

Отклонения описываются нормальным законом распределения случайных величин. При этом наследственные признаки закрепляются, если они имеют приспособительный характер, т. е. обеспечивают популяции лучшие условия существования и размножения.

Так же как процесс эволюции начинается с начальной популяции, так и алгоритм начинает свою работу с создания начального множества конкурирующих между собой решений оптимизационной задачи. Затем эти «родительские» решения создают «потомков» путем случайных и направленных изменений. После этого оценивается эффективность этих решений, и они подвергаются селекции. Аналогично естественным системам здесь действует принцип «выживания сильнейших», наименее приспособленные решения «погибают», а затем процесс повторяется вновь и вновь.

Традиционные оптимизационные алгоритмы для нахождения лучшего решения используют большое количество допущений при оценке целевой функции. Эволюционный подход не требует таких допущений, что расширяет класс задач, которые можно решать с помощью генетических алгоритмов. Согласно существующим исследованиям можно сказать, что генетические алгоритмы позволяют решать те проблемы, решение которых традиционными алгоритмами затруднительно.

Генетический алгоритм дает преимущества при решении практических задач. Одно из них — это адаптация к изменяющейся окружающей среде. В реальной жизни проблема, которая была поставлена для решения изначально, может претерпеть огромные изменения в процессе своего решения. При использовании традиционных методов все вычисления приходится начинать заново, что приводит к большим затратам машинного времени. При эволюционном подходе популяцию можно анализировать, дополнять и видоизменять применительно к изменяющимся условиям, для этого не требуется полный перебор. Другое преимущество генетических алгоритмов для решения задач состоит в способности быстрой генерации достаточно хороших решений.

3. При решении практических задач с использованием генетических алгоритмов обычно выполняют четыре предварительных этапа:

- выбор способа представления решения;
- разработка операторов случайных изменений;
- определение способов «выживания» решений;
- создание начальной популяции альтернативных решений.

Рассмотрим некоторые особенности выполнения этих этапов.

На первом этапе для представления решения в формальном виде требуется такая структура, которая позволит кодировать любое возможное решение и производить его оценку. Математически доказано, что не существует идеальной структуры представления, так что для создания хорошей структуры требуется анализ, перебор и эвристические подходы. Возможный вариант представления должен позволять

проведение различных перестановок в альтернативных решениях. Для оценки решений необходимо определить способ вычисления целевой функции.

На втором этапе достаточно сложным является выбор случайного оператора (или операторов) для генерации потомков. Существует огромное число таких операторов. Как отмечалось в главе 1, существуют два основных типа размножения: половое и бесполое. При половом размножении два родителя обмениваются генетическим материалом, который используется при создании потомка. Бесполое размножение — это фактически клонирование, при котором происходят различные мутации при передаче информации от родителя к потомку. Модели этих типов размножения играют важную роль в генетических алгоритмах. В общем случае можно применить модели размножения, которые не существуют в природе. Например, использовать материал от трех или более родителей, проводить голосование при выборе родителей и т. п. Фактически нет пределов в использовании различных моделей, и поэтому при решении технических задач нет смысла слепо копировать законы природы и ограничиваться только ими.

Успех генетических алгоритмов во многом зависит от того, как взаимодействуют между собой схема представления, методы случайных изменений и способ определения целевой функции. Поэтому для определенного класса задач целесообразно использовать направленные методы.

В качестве примера рассмотрим два способа представления перестановок при решении оптимизационных задач. В первом случае будем использовать одного родителя (альтернативное решение) и получать одного потомка. Во втором случае используем двух родителей, случайно выберем точку перестановки и для образования потомка возьмем первый сегмент у первого родителя, а второй сегмент — у второго. Первый метод похож на бесполое размножение, а второй — на половое размножение. Стоит отметить, что если первый метод всегда генерирует реальное решение, то второй может генерировать недопустимые решения. При этом требуется «восстанавливать» допустимые решения перед их оценкой.

На третьем из рассматриваемых этапов задаются правила выживания решений для создания потомства. Существует множество способов проведения селекции альтернативных решений. Простейшее правило — это «выживание сильнейших», т. е. остаются только лучшие решения с точки зрения заданной целевой функции, а все остальные устраняются. Такое правило часто оказывается малоэффективным при решении сложных технических проблем. Иногда лучшие решения могут происходить от худших, а не только от самых лучших. Тем не менее, логично использовать принцип:

Чем «лучше» решение, тем больше вероятность его выживания

Отметим, что ***принцип*** (от латинского «начало») — это:
- основное исходное положение какой-либо теории;
- внутренняя убежденность в чем-либо;
- основная особенность работы механизма, устройства и т. п.

На последнем предварительном этапе создается начальная популяция. При неполноте исходных данных о проблеме решения могут случайным образом выбираться из всего множества альтернатив. Это реализуется генерацией случайных внутрихромосомных перестановок, каждая из которых представляет собой определенное решение. При создании начальной популяции рекомендуется использовать знания о решаемой задаче. Например, эти знания могут быть получены из опыта разработчика, существующих стандартов и библиотек алгоритмов решения задач данного класса.

4. ***Эффективность генетического алгоритма*** — степень реализации запланированных действий алгоритма и достижение требуемых значений целевой функции. Эффективность во многом определяется структурой и составом начальной популяции. При создании начального множества решений происходит формирование популяции на основе четырех основных принципов:
- «одеяло» — генерируется полная популяция, включающая все возможные решения в некоторой заданной области;
- «дробовик» — подразумевает случайный выбор альтернатив из всей области решений данной задачи.
- «фокусировка» — реализует случайный выбор допустимых альтернатив из заданной области решений данной задачи.
- «комбинирование» — состоит в различных совместных реализациях первых трех принципов.

Отметим, что популяция обязательно является конечным множеством.

3.2. Генетические операторы

1. В каждой генерации генетического алгоритма хромосомы являются результатом применения некоторых генетических операторов.

Оператор — это языковая конструкция, представляющая один шаг из последовательности действий или набора описаний алгоритма.

Генетический алгоритм состоит из набора генетических операторов.

Генетический оператор по аналогии с оператором алгоритма — средство отображения одного множества на другое. Другими словами, это конструкция, представляющая один шаг из последовательности действий генетического алгоритма.

2. Рассмотрим основные операторы генетических алгоритмов.

Оператор репродукции (***селекция***) — это процесс, посредством которого хромосомы (альтернативные решения), имеющие более высокое значение целевой функции (с «лучшими» признаками), получают

бóльшую возможность для воспроизводства (репродукции) потомков, чем «худшие» хромосомы. Элементы, выбранные для репродукции, обмениваются генетическим материалом, создавая аналогичных или различных потомков.

Существует большое число видов операторов репродукции. К ним относятся следующие.

- *Селекция на основе рулетки* — это простой и широко используемый в простом генетическом алгоритме метод. При его реализации каждому элементу в популяции соответствует зона на колесе рулетки, пропорционально соразмерная с величиной целевой функции. Тогда при повороте колеса рулетки каждый элемент имеет некоторую вероятность выбора для селекции. Причем элемент с большим значением целевой функции имеет большую вероятность для выбора.
- *Селекция на основе заданной шкалы*. Здесь популяция предварительно сортируется от «лучшей» к «худшей» на основе заданного критерия. Каждому элементу назначается определенное число и тогда селекция выполняется согласно этому числу.
- *Элитная селекция*. В этом случае выбираются лучшие (элитные) элементы на основе сравнения значений целевой функции. Далее они вступают в различные преобразования, после которых снова выбираются элитные элементы. Процесс продолжается аналогично до тех пор, пока продолжают появляться элитные элементы.
- *Турнирная селекция*. При этом некоторое число элементов (согласно размеру «турнира») выбирается случайно или направленно из популяции, и лучшие элементы в этой группе на основе заданного турнира определяются для дальнейшего эволюционного поиска.

Оператор репродукции считается эффективным, если он создает возможность перехода из одной подобласти альтернативных решений области поиска в другую. Это повышает вероятность нахождения глобального оптимума целевой функции. Выделяют два основных типа реализации оператора репродукции:

- случайный выбор хромосом;
- выбор хромосом на основе значений целевой функции.

При случайном выборе хромосом частота R образования родительских пар не зависит от значения целевой функции хромосом P_k^t и полностью определяется численностью популяции N:

$$R = \frac{\beta}{N(N-1)}, \tag{3.1}$$

где β — коэффициент селекции, принимающий в зависимости от условий внешней среды значение $1 \div 4$.

Другой способ реализации оператора репродукции связан с использованием значений целевой функции. Существуют две основные стратегии. ***Стратегия*** — это оптимальный набор правил и приемов, которые позволяют реализовать общую цель, достигнуть глобальных

и локальных целей решаемой задачи. В первой предпочтение отдается хромосомам с близкими и «лучшими» (наибольшими при максимизации и наименьшими — при минимизации) значениями целевой функции. Во второй — хромосомам, со значениями целевой функции, сильно различающимися между собой.

Для реализации первой стратегии с максимизацией целевой функции с вероятностью

$$\Pr(\text{ОР}) = \frac{\beta}{\text{ЦФ}(P_k^t)} \Big/ \sum_{i=1}^{\text{N}} \text{ЦФ}(P_k^t), \quad k = \overline{1, N}, \tag{3.2}$$

выбирают разные хромосомы. Здесь ЦФ — целевая функция, ОР — это оператор репродукции, моделирующий естественный процесс селекции, Pr (OP) — вероятность выбора хромосом для репродукции.

Вторая стратегия реализуется так: часть хромосом выбирается случайным образом, а вторая — с вероятностью на основе выражения (3.2).

Если $\text{ЦФ}(P_k^t) < \text{ЦФ}_{\text{ср}}$, где $\text{ЦФ}_{\text{ср}}$ — среднее значение целевой функции в популяции, то оператор репродукции моделирует естественный отбор. Выбор случайных и сильно отличающихся хромосом повышает генетическое разнообразие популяции, что повышает скорость сходимости генетического алгоритма на начальном этапе оптимизации и позволяет в некоторых случаях выходить из локальных оптимумов.

3. Кроме описанных, существует большое число других методов селекции, которые можно условно классифицировать на три группы. К первой группе отнесем вероятностные методы. Ко второй — детерминированные методы. К третьей — различные комбинации методов из первой и второй групп. Построение новых операторов репродукции непрерывно продолжается.

4. Основной трудностью решения инженерных оптимизационных задач с большим количеством локальных оптимумов является ***предварительная сходимость алгоритмов***. Другими словами, попадание решения в один, далеко не самый лучший, локальный оптимум при наличии их большого количества. Различные методы селекции и их модификации как раз и позволяют в некоторых случаях решать проблему предварительной сходимости алгоритмов. Следует отметить, что исследователи генетических алгоритмов все более склоняются к мысли применять комбинированные методы селекции с использованием предварительных знаний о решаемых задачах и предварительных результатах.

5. Опишем теперь ***операторы кроссинговера*** (***скрещивания***). ***Оператор кроссинговера*** — это языковая конструкция, позволяющая на основе преобразования (скрещивания) хромосом родителей (или их частей) создавать хромосомы потомков. Существует огромное число опе-

раторов кроссинговера, так как их структура в основном и определяет эффективность генетических алгоритмов. Кратко рассмотрим основные операторы кроссинговера, известные в литературе, и их модификации.

5.1. ***Простой*** **(*одноточечный*)** ***оператор кроссинговера***. Перед началом работы одноточечного оператора кроссинговера определяется так называемая точка оператора кроссинговера, или разрезающая точка оператора кроссинговера, которая обычно определяется случайно. Эта точка определяет место в двух хромосомах, где они должны быть «разрезаны». Например, пусть популяция P состоит из хромосом P_1 и P_2, которые выступают в качестве родителей, $P = \{P_1, P_2\}$. Пусть первый и второй родители имеют вид $P_1 : 11111$, $P_2 : 00000$. Выберем точку оператора кроссинговера между вторым и третьим генами в P_1, P_2. Тогда, меняя элементы после точки оператора кроссинговера между двумя родителями, можно создать два новых потомка. В нашем примере получим

$$\begin{array}{lccc} P_1: & 1\ 1 & | & 1\ 1\ 1 \\ P_2: & 0\ 0 & | & 0\ 0\ 0 \\ \hline P_1': & 1\ 1 & | & 0\ 0\ 0 \\ P_2': & 0\ 0 & | & 1\ 1\ 1 \end{array}$$

Итак, одноточечный оператор кроссинговера выполняется в три этапа:

1. Две хромосомы $A = a_1, a_2, \ldots, a_L$ и $B = a'_1, a'_2, \ldots, a'_L$ выбираются случайно из текущей популяции.

2. Число k выбирается из $\{1, 2, \ldots, L-1\}$ также случайно. Здесь L — длина хромосомы, k — точка оператора кроссинговера (номер, значение или код гена, после которого выполняется разрез хромосомы).

3. Две новые хромосомы формируются из A и B путем перестановок элементов согласно правилу

$$A' = a_1, a_2, \ldots, a_k, a'_{k+1}, \ldots, a'_L,$$

$$B' = a'_1, a'_2, \ldots, a'_k, a_{k+1}, \ldots, a_L.$$

После применения оператора кроссинговера имеем две старые хромосомы и всегда получаем две новые хромосомы. Схематически простой оператор кроссинговера показывает преобразование двух хромосом и частичный обмен информацией между ними, использующий точку разрыва, выбранную случайно.

5.2. ***Двухточечный оператор кроссинговера.*** В каждой хромосоме определяются две точки оператора кроссинговера, и хромосомы обмениваются участками, расположенными между двумя точками оператора кроссинговера. Например:

$$\begin{array}{lccccc} P_1: & 1\ 1\ 1 & | & 0\ 1 & | & 0\ 0 \\ P_2: & 0\ 0\ 0 & | & 1\ 1 & | & 1\ 0 \\ \hline P_1': & 1\ 1\ 1 & | & 1\ 1 & | & 0\ 0 \\ P_2': & 0\ 0\ 0 & | & 0\ 1 & | & 1\ 0 \end{array}$$

Отметим, что точки оператора кроссинговера в двухточечном операторе кроссинговера также определяются случайно. Существует большое количество модификаций двухточечного оператора кроссинговера. Развитием двухточечного оператора кроссинговера является многоточечный или N-точечный оператор кроссинговера. Многоточечный оператор кроссинговера выполняется аналогично двухточечному, хотя большое число «разрезающих» точек может привести к потере «хороших» родительских свойств.

Пример трехточечного оператора кроссинговера:

$$\begin{array}{lcccccccccc} P_1: & 1 & | & 1 & 1 & | & 0 & 1 & | & 0 & 0 \\ P_2: & 0 & | & 0 & 0 & | & 1 & 0 & | & 1 & 1 \\ \hline P_1': & 1 & | & 0 & 0 & | & 0 & 1 & | & 1 & 1 \\ P_2': & 0 & | & 1 & 1 & | & 1 & 0 & | & 0 & 0 \end{array}$$

Здесь точки оператора кроссинговера делят хромосому на ряд строительных блоков (в данном случае 4). Потомок P_1' образуется из нечетных блоков родителя P_1 и четных блоков родителя P_2. Потомок P_2' образуется соответственно из нечетных блоков родителя P_2 и четных блоков родителя P_2.

Тогда многоточечный оператор кроссинговера выполняется аналогичным образом.

5.3. ***Упорядоченный оператор кроссинговера***. Здесь «разрезающая» точка также выбирается случайно. Далее происходит копирование левого сегмента P_1 в P_1'. Остальные позиции в P_1' берутся из P_2 в упорядоченном виде слева направо, исключая элементы, уже попавшие в P_1'. Например:

$$\begin{array}{lcccccccccc} P_1: & A & B & C & D & | & E & F & G & H \\ P_2: & G & A & B & E & | & C & D & F & H \\ \hline P_1': & A & B & C & D & | & G & E & F & H \end{array}$$

Получение P_2' может выполняться различными способами. Наиболее распространенный метод копирования левого сегмента из P_2, а далее анализ P_1 методом, указанным выше. Тогда имеем P_2' : $(G\ A\ B\ E \mid C\ D\ F\ H)$.

5.4. ***Частично-соответствующий оператор кроссинговера***. Здесь также случайно выбирается «разрезающая» точка или точка оператора кроссинговера. Дальше анализируются сегменты (части) в обеих хромосомах и устанавливается частичное соответствие между элементами первого и второго родителей с формированием потомков. При этом правый сегмент P_2 переносится в P_1', левый сегмент P_1 переносится в P_1' с заменой повторяющихся генов на отсутствующие гены, находящиеся в частичном соответствии. Например:

$$\begin{array}{lcccccccccc} P_1: & A & B & C & D & E & F & | & G & H & I & J \\ P_2: & E & C & I & A & D & H & | & J & B & F & G \\ \hline P_1': & A & H & C & D & E & I & | & J & B & F & G \end{array}$$

Аналогично можно получить P_2':

P_1:	A	B	C	D	E	F	|	G	H	I	J
P_2:	E	C	I	A	D	H	|	J	B	F	G
P_2':	E	C	F	A	D	B	|	G	H	I	J

5.5. ***Циклический оператор кроссинговера***. Циклический оператор кроссинговера выполняет рекомбинации согласно циклам, которые существуют при установлении соответствия между генами первого и второго родителей. Например, пусть популяция P состоит из двух хромосом: P_1 и P_2, $P = \{P_1, P_2\}$. Первый и второй родители и их потомок имеют вид:

P_1:	1	2	3	4	5	6	7	8	9	10
P_2:	5	3	9	1	4	8	10	2	6	7
P_1':	1	3	9	4	5	8	10	2	6	7

При выполнении циклического оператора кроссинговера P_1' заполняется, начиная с первой позиции, и копирует элемент с первой позиции P_1. Элементу 1 в P_1 соответствует элемент 5 в P_2. Следовательно, имеем первый путь в цикле (1,5). Элементу 5 в P_1 соответствует элемент 4 в P_2, откуда второй путь в первом цикле (1,5; 5,4). Продолжая далее, получим, что элементу 4 в P_1 соответствует элемент 1 в P_2. Следовательно, сформирован первый цикл (1,5; 5,4; 4,1). Согласно этому циклу элемент 5 переходит в пятую позицию P_1', а элемент 4 — в четвертую позицию.

Сформируем теперь второй цикл. Элемент 2 в P_1 соответствует элементу 3 в P_2. Продолжая аналогично, получим второй цикл (2,3; 3,9; 9,6; 6,8; 8,2) и третий (7,10; 10,7) циклы.

Следовательно, в P_1' элемент 3 расположен во втором локусе, т. е. на второй позиции, элемент 9 — в третьем, элемент 6 — в девятом, элемент 8 — в шестом, элемент 2 — в восьмом, элемент 10 — в седьмом и элемент 7 — в десятом локусах. Циклический оператор кроссинговера и его модификации эффективно применяются для решения комбинаторно-логических задач, задач на графах и гиперграфах и других оптимизационных задач.

5.6. ***Универсальный оператор кроссинговера***. В настоящее время он популярен для решения различных задач из теории расписаний. Вместо использования разрезающей точки (точек) в универсальный оператор кроссинговера вводят двоичную маску, длина которой равна длине заданных хромосом. Первый потомок получается сложением первого родителя с маской на основе следующих правил: $(0 + 0 = 0,\ 0 + 1 = 1,\ 1 + 1 = 0)$. Второй потомок получается анало-

гичным образом. Для каждого элемента в P_1, P_2 гены меняются, как показано на следующем примере:

P_1 :	0	1	1	0	0	1	
P_2 :	0	1	0	1	1	1	
	0	1	1	0	1	0	— маска
P_1' :	0	0	0	0	1	1	
P_2' :	0	0	1	1	0	1	

Маска может быть задана или выбирается случайно с заданной вероятностью или на основе генератора случайных чисел. При этом чередование 0 и 1 в маске происходит с вероятностью $\approx 50\,\%$. В некоторых случаях используется параметризированный универсальный оператор кроссинговера, где маска может выбираться с вероятностью для 1 или 0 выше, чем 50 %. Такой вид маски эффективен, когда хромосомы кодируются в двоичном алфавите.

5.7. Для решения многих оптимизационных задач можно использовать некоторые классы алгоритмов, называемых «***жадными***». Такой алгоритм делает на каждом шаге локально оптимальный выбор, в надежде, что итоговое решение также окажется оптимальным. Это не всегда так, но для многих задач такие эвристические алгоритмы дают оптимальный результат. Говорят, что к оптимизационной задаче применим принцип «жадного» выбора, если последовательность локально-оптимальных («жадных») выборов дает оптимальное решение.

«***Жадный» оператор*** — это языковая конструкция, позволяющая создавать новые решения на основе частичного выбора на каждом шаге преобразования локально оптимального значения целевой функции.

Рассмотрим «***жадный» оператор кроссинговера***. Он может быть реализован на двух и более хромосомах, а в пределе — на всей популяции. Приведем алгоритм «жадного» оператора кроссинговера на примере нахождения пути с минимальной или максимальной стоимостью на графе:

1. Для всех хромосом популяции вычисляется целевая функция (стоимость пути между всеми вершинами графа). Выбирается заданное число родительских хромосом и случайным образом на одной из хромосом определяется точка «жадного» оператора кроссинговера.
2. В выбранной хромосоме для i-го гена, расположенного слева от точки «жадного» оператора кроссинговера, определяется значение частичной целевой функции. Например, это стоимость пути от выбранного гена к ближайшему, находящемуся справа гену. Аналогичные действия выполняются по определению стоимости пути от i-го гена во всех остальных хромосомах, выбранных для «жадного» оператора кроссинговера.
3. В хромосому «потомок» выбирают тот ген, у которого значение целевой функции выше (ниже) при максимизации (минимизации) значений целевой функции.

4. Процесс продолжается аналогично, до построения хромосомы «потомок». Если в процессе реализации возникает цикл или тупик, то выбираются нерассмотренные гены с лучшим значением целевой функции.

Например, пусть задан граф $G = (X, U)$, $X = \{a, b, c, d, e\}$ в виде матрицы. Построим популяцию P, состоящую из трех родительских хромосом $P = \{P_1, P_2, P_3\}$, где $P_1 : a\,b\,c\,d\,e$; $P_2 : b\,d\,e\,c\,a$; $P_3 : e\,b\,a\,d\,c$. Элементы матрицы определяют стоимость пути между любыми двумя вершинами графа, а каждый ген в хромосоме кодируется номером вершины графа:

	a	b	c	d	e
a	–	15	6	7	8
b	15	–	4	3	2
c	6	4	–	1	10
d	7	3	1	–	9
e	8	2	10	9	–

Согласно алгоритму выберем точку «жадного» оператора кроссинговера между генами b и c в хромосоме P_1. Теперь выбор $(b - c)$ дает значение целевой функции, равное 4 (см. матрицу), выбор $(b - d)$ (в хромосоме P_2) определяет целевую функцию со значением 3, а выбор $(b - a)$ (в хромосоме P_3) определяет целевую функцию, равную 15. При минимизации целевой функции выберем путь $(b - d)$. Продолжая, получим путь реализации «жадного» оператора кроссинговера (рис. 3.1). Итак, хромосома потомка $P' : b\,d\,c\,a\,e$ имеет суммарную целевую функцию, равную 18: $3 + 1 + 6 + 8 = 18$, а целевая функция родителей для P_1 равна $15 + 4 + 1 + 9 = 29$, для P_2 равна $3 + 9 + 10 + 6 = 28$ и для P_3 равна $2 + 15 + 7 + 1 = 25$.

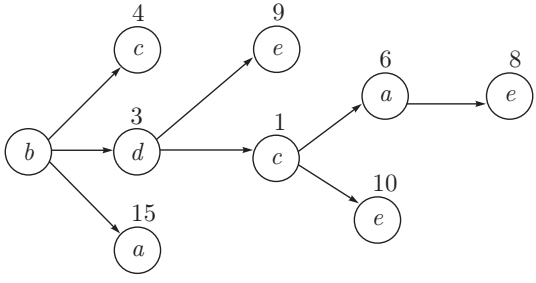


Рис. 3.1. Пример реализации «жадного» оператора кроссинговера

Стратегию «жадного» оператора кроссинговера можно выполнять различными способами. Следует отметить, что поиск универсальных и специализированных операторов кроссинговера, ориентированных на решение заданного класса задач, продолжается.

Рассмотрим различные варианты выбора пары хромосом для скрещивания. Вероятность скрещивания $P_{л,\,x}(\text{ОК})$ лучших хромосом с худшими по значению целевой функции, должна уменьшаться при эволюции поколений: ОК — оператор кроссинговера.

Вероятность скрещивания лучших хромосом должна увеличиваться на последних этапах оператора кроссинговера для закрепления желаемых признаков в хромосомах.

5.8. Существуют и другие методы выбора пар хромосом для скрещивания. Например, «близкое» родство, «дальнее» родство, выбор на основе кода Грея и т. д.

«Близкое родство». Здесь вероятность выбора хромосом, подлежащих скрещиванию, запишется так:

— для первой хромосомы P_i

$$P_{\text{бр},\,i}(\text{ОК}) = \sqrt{P_{\text{бр},\,o}(\text{ОК})} \cdot \left(1 - \exp\left(-\frac{1}{t}\alpha\right)\right), \qquad (3.3)$$

где $P_{\text{бр},\,o}(\text{ОК})$ — вероятность выбора хромосомы на основе близкого родства;

— для второй хромосомы P_j вероятность $P_{\text{бр},\,j}(\text{ОК})$ задается пользователем.

Затем вычисляется ***Хэммингово расстояние*** $\text{dist}(Pi, Pj)$ между выбранными хромосомами текущей популяции. Рассмотрим, например, две хромосомы, каждая из которых задается конкретными значениями семи переменных: x_1, x_2, x_3, x_4, x_5, x_6, x_7; пусть в первом случае ими будут 0, 1, 0, 1, 1, 0, 1, а во втором 1, 1, 0, 1, 0, 0, 0. Запишем данные совокупности значений $x_1, x_2, \ldots, x_7$ одну под другой:

x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	1	0	1	1	0	1
1	1	0	1	0	0	0
–1	0	0	0	1	0	1

Вычислим разность между каждым элементом первой строки и соответствующим элементом второй строки. Запишем результаты под второй строкой. Теперь вычислим сумму «абсолютных значений» полученных результатов. Эта сумма равна 3. Говорят, что расстояние Хэмминга между указанными выше точками (или величинами) [0, 1, 0, 1, 1, 0, 1] и [1, 1, 0, 1, 0, 0, 0] равно 3:

$$|-1| + |1| + |1| = 1 + 1 + 1 = 3.$$

В общем случае рассмотрим две хромосомы $(x'_1, x'_2, \ldots, x'_n)$ и $(x_1, x_2, \ldots, x_n)$; обобщенным расстоянием Хэмминга между этими двумя хромосомами называется скаляр

$$\pi = |x'_1 - x_1| + |x'_2 - x_2| + \ldots + |x'_n - x_n|.$$

Хромосомы рекомендуются для скрещивания, если хеммингово расстояние между ними $\text{dist}(Pi, Pj) > R$, где R — радиус скрещива-

ния, задаваемый лицом, принимающим решение, или определяется как ближайшее меньшее целое от разности значений целевых функций ЦФ(Pi) – ЦФ(Pj), деленное на два.

Вероятности $P_{\text{бр},\,i}(\text{ОК})$ и $P_{\text{бр},\,j}(\text{ОК})$ возрастают на конечных стадиях работы оператора кроссинговера.

«Дальнее» родство определяется условием

$$P_{\text{др}}(\text{ОК}) = \sqrt{P_{\text{бр},\,o}(\text{ОК})} \cdot \exp\left(-\frac{1}{t}\alpha\right),$$

где $P_{\text{др},\,i}(\text{ОК})$ — вероятность выбора хромосом при «дальнем» родстве на начальных стадиях работы генетических алгоритмов, с учетом особенностей вычисляется для первой и второй хромосомы.

Хромосомы P_i и P_j подлежат скрещиванию, если хеммингово расстояние между ними $\text{dist}(Pi, Pj) > R$. Вероятность $P_{\text{др},\,i}(\text{ОК})$ и $P_{\text{др},\,j}(\text{ОК})$ уменьшается на конечных стадиях поиска оптимального решения.

Код Грея — это двоичный код, последовательные значения которого отличаются только одним двоичным разрядом. Код Грея может использоваться для хромосом, представленных в двоичном виде. Например:

0 — 0000
1 — 0001
2 — 0011
3 — 0010
4 — 0110
5 — 0111 и т. д.

Такое кодирование альтернативных решений позволяет решать вопросы «взбалтывания» популяции; оно эффективно на начальных стадиях генетического алгоритма.

Как следует из биологии, некоторые процессы преобразования популяции происходят толчками. Основой таких процессов являются точковые мутации. В генетическом алгоритме мутация необходима потому, что предотвращают потерю генетического материала. Точковые мутации не изменяют размера и строения хромосом, а изменяют взаимное расположение генов в хромосоме.

5.9. ***Оператор мутации*** — языковая конструкция, позволяющая на основе преобразования родительской хромосомы (или ее части) создавать хромосому потомка.

Оператор мутации обычно состоит из двух этапов:

1. В хромосоме $A = (a_1, a_2, a_3, \ldots, a_{L-2}, a_{L-1}, a_L)$ определяются случайным образом две позиции (например, a_2 и a_{L-1}).
2. Гены, соответствующие выбранным позициям, переставляются, и формируется новая хромосома $A' = (a_1, a_{L-1}, a_3, \ldots, a_{L-2}, a_2, a_{L)}$.

Рассмотрим кратко основные операторы мутации. Простейшим оператором мутации является одноточечный. При его реализации случай-

но выбирают ген в родительской хромосоме и, обменивая его на рядом расположенный ген, получают хромосому потомка, например:

$$\begin{array}{ll} P_1: & 0\ 1\ 1\ |\ 0\ 1\ 1 \\ P_1': & 0\ 1\ 0\ |\ 1\ 1\ 1 \end{array}$$

Здесь P_1 — родительская хромосома, а P_1' — хромосома-потомок после применения одноточечного оператора мутации.

При реализации ***двухточечного оператора мутации*** случайным или направленным образом выбираются две точки разреза. Затем производится перестановка генов между собой, расположенных справа от точек разреза, например:

$$\begin{array}{ll} P: & A\ |\ B\ C\ D\ |\ E\ F \\ P': & A\ \ \ E\ C\ D\ \ \ B\ F \end{array}$$

Здесь P_1 — родительская хромосома, а P_1' — хромосома-потомок после применения двухточечного оператора мутации.

Развитием двухточечного оператора мутации является многоточечный (или n-точечный) оператор мутации. В этом случае происходит последовательный обмен генов, расположенных правее точек разреза друг с другом в порядке их расположения. Ген, расположенный правее последней точки разреза, переходит на место первого, например:

$$\begin{array}{ll} P: & A\ |\ B\ C\ D\ |\ E\ F\ |\ G\ H \\ P': & A\ \ \ G\ C\ D\ \ \ B\ F\ \ \ E\ H \end{array}$$

Строительные блоки — это тесно связанные между собой гены (части альтернативных решений), которые нежелательно изменять при выполнении генетических операторов. Из строительных блоков (как из кирпичиков при построении дома) можно создавать альтернативные оптимальные или квазиоптимальные решения.

В частном случае, когда строительные блоки, расположенные между точками разреза, одинаковы, многоточечный оператор мутации выполняется следующим образом. При четном числе точек разреза меняются местами гены, расположенные справа и слева от выбранных точек, например:

$$\begin{array}{ll} P: & A\ B\ |\ C\ D\ |\ E\ F\ |\ G\ H\ |\ I\ J \\ P': & A\ C\ \ \ B\ E\ \ \ D\ G\ \ \ F\ I\ \ \ H\ J \end{array}$$

При нечетном числе точек потомок получается после обмена участками хромосом, расположенных между четными точками разреза, например:

$$\begin{array}{ll} P: & A\ B\ C\ |\ D\ E\ F\ |\ G\ H\ I\ |\ J \\ P': & A\ B\ C\ \ \ G\ H\ I\ \ \ D\ E\ F\ \ \ J \end{array}$$

Часто используют операторы мутации, использующие знания о решаемой задаче. Реализация таких операторов заключается в переста-

новке местами любых выбранных генов в хромосоме, причем точка или точки мутации определяются не случайно, а направленно.

В позиционном операторе мутации две точки разреза выбираются случайно, а затем ген, соответствующий второй точке мутации, размещается в позицию перед геном, соответствующим первой точке, например:

$$\begin{array}{lcccccccc} P: & A & | & B & C & D & | & E & F \\ P': & A & & E & B & C & & D & F \end{array}$$

5.10. Введем понятие оператора инверсии. ***Оператор инверсии*** — это языковая конструкция, позволяющая на основе инвертирования родительской хромосомы (или ее части) создавать хромосому потомка. При его реализации случайным образом определяется одна или несколько точек разреза (инверсии), внутри которых элементы инвертируются.

Генетический оператор инверсии состоит из следующих шагов.

1. Хромосома $B = b_1, b_2, \ldots, b_L$ выбирается случайным образом из текущей популяции.
2. Два числа y_1' и y_2' выбираются случайным образом из множества $\{0, 1, 2, \ldots, L+1\}$, причем считается, что $y_1 = \min\{y_1', y_2'\}$ и $y_2 = \max\{y_1', y_2'\}$.
3. Новая хромосома формируется из B путем инверсии сегмента, который лежит справа от позиции y_1 и слева от позиции y_2 в хромосоме B. Тогда после применения оператора инверсии получаем:

$$B_1 = (b_1, \ldots, b_{y_1}, b_{y_2-1}, b_{y_2-2}, \ldots, b_{y_1+1}, b_{y_2}, \ldots, b_L).$$

Для одноточечного оператора инверсии запишем

$$\begin{array}{lcccccccc} P_2: & A & | & B & C & D & E & F & G & H \\ P_2': & A & | & H & G & F & E & D & C & B \end{array}$$

Здесь P_2 — родительская хромосома, P_2' — хромосома-потомок после применения оператора инверсии.

Например, для двухточечного оператора инверсии получим

$$\begin{array}{lcccccccccc} P_1: & A & B & C & | & D & E & F & | & G & H \\ P_1': & A & B & C & | & F & E & D & | & G & H \end{array}$$

Здесь P_1 — родительская хромосома, P_1' — хромосома-потомок после применения двухточечного оператора инверсии.

Часто применяется специальный оператор инверсии. В нем точки инверсии определяются с заданной вероятностью для каждой новой создаваемой хромосомы в популяции.

5.11. Рассмотрим оператор транслокации. ***Оператор транслокации*** — это языковая конструкция, позволяющая на основе скрещивания и инвертирования из пары родительских хромосом (или их частей) создавать две хромосомы потомков. Другими словами, он представляет

собой комбинацию операторов кроссинговера и инверсии. В процессе его реализации случайным образом производится один разрез в каждой хромосоме. При формировании потомка P_1' берется левая часть до разрыва из родителя P_1 и инверсия правой части до разрыва из P_2. При создании P_2' берется левая часть P_2 и инверсия правой части P_1, например:

$$\begin{array}{lcccccccc} P_1: & A & B & | & C & D & E & F \\ P_2: & G & K & | & H & I & J & Q \\ \hline P_1': & A & B & & Q & J & I & H \\ P_2': & G & K & & F & E & D & C \end{array}$$

Существует большое число других видов оператора транслокации. Отметим, что до последнего времени оператор транслокации не применялся в генетических алгоритмах, а также при разработке интеллектуальной искусственной системы и решении оптимизационных задач.

5.12. ***Оператор транспозиции*** — языковая конструкция, позволяющая на основе преобразования и инвертирования выделяемой части родительской хромосомы создавать хромосому потомка. Например,

$$\begin{array}{lccccccccccc} P_1: & A & | & B & C & | & D & E & F & | & G & H \\ P_1^1: & A & & F & E & & D & B & C & & G & H \end{array}$$

Здесь три точки разреза. Точки разреза выбираются случайным или направленным образом. В родительской хромосоме P_1 строительный блок DEF инвертируется и вставляется в точку разреза между генами A и B. В результате получаем хромосому-потомок P_1^1. Отметим, что существует большое количество модификаций оператора транспозиции.

5.13. Рассмотрим ***оператор сегрегации***. Это языковая конструкция, позволяющая на основе выбора строительных блоков из хромосом родителей (или их частей) создавать хромосомы потомков.

Приведем один из примеров его реализации. Отметим, что оператор сегрегации, как правило, реализуется на некотором наборе хромосом. Пусть имеется популяция P, состоящая из четырех родительских хромосом:

$$P = \{P_1, P_2, P_3, P_4\}$$
$$P_1 : (12345678); P_2 : (24316587); P_3 : (31425768); P_4 : (87654321).$$

В каждой хромосоме выделим строительные блоки. Выделим по два строительных блока: в P_1 — 23 и 67, в P_2 — 1 и 4, в P_3 — 768 и 425 и в P_4 — 54 и 87. Тогда, например, потомок P_1' можно сформировать, взяв первые строительные блоки из каждой родительской хромосомы популяции. Так, вариант P_1' будет представлен последовательностью (23176854) и является реальным решением, а вариант P_2' (67442587) является нереальным (недопустимым). Очевидно, что оператор сегрегации можно реализовать различными способами в зависимости от выборки строительный блоков или генов из хромосом.

5.14. Опишем ***оператор удаления***. Это языковая конструкция, позволяющая на основе удаления строительных блоков из хромосом родителей (или их частей) создавать хромосомы потомков.

При его реализации направленным или случайным образом определяется точка или точки разреза. Далее производится пробное удаление генов или их строительных блоков с вычислением изменения значения целевой функции. Элементы, расположенные справа от точки оператора удаления или между двумя точками, удаляются из хромосомы. При этом производится преобразование потомка таким образом, чтобы соответствующее альтернативное решение оставалось реальным.

5.15. Опишем ***оператор вставки***. Это языковая конструкция, позволяющая на основе вставки строительных блоков в хромосомы родителей создавать хромосомы потомков.

При его реализации направленным или случайным образом создается хромосома (донор), состоящая из строительных блоков, которые желательно разместить в другие хромосомы популяции. После этого направленным или случайным образом определяется хромосома для реализации оператора вставки. В ней находится точка или точки разреза. Затем анализируются другие хромосомы в популяции для определения альтернативных вставок. Далее производится пробная вставка строительных блоков с вычислением изменения значения целевой функции и получением реальных решений. Новые строительные блоки вставляются в хромосому справа от точки оператора вставки или между его двумя точками. Отметим, что оператор удаления и оператор вставки могут изменять размер хромосом. Для сохранения постоянного размера хромосом эти операторы можно применять совместно.

На конечном этапе поиска целесообразно применять выбор близких решений в соответствии с определенным критерием, т. е. искать решение среди лучших P_k^t.

5.16. ***Оператор редукции*** — языковая конструкция, позволяющая на основе анализа популяции после одной или нескольких поколений генетического алгоритма уменьшать ее размер до заданной величины. Рассмотрим способы реализации оператора редукции. Он выполняется для устранения неудачных решений. В некоторых генетических алгоритмах, в частности, в простом генетическом алгоритме, этот оператор применяется для сохранения постоянного размера начальной популяции. Основная проблема здесь — нахождение компромисса между разнообразием генетического материала и качеством решений. Сначала формируют репродукционную группу из всех решений, образовавшихся в популяции P^t, затем выполняют отбор решений в следующую популяцию.

Численность новой популяции

$$N^{t+1} = N^t + N_{\text{ок}} + N_{\text{ом}} + N_{\text{ои}} + N_{\text{от}} + N_{\text{отр}} + N_{\text{ос}} + N_{\text{оу}} + N_{\text{ов}},$$

где
N^{t+1} — численность новой популяции,
N^t — численность популяции на предыдущем шаге (поколении) t,
$N_{\text{ок}}$, $N_{\text{ом}}$, $N_{\text{от}}$, $N_{\text{ои}}$, $N_{\text{ос}}$, $N_{\text{отр}}$ $N_{\text{оу}}$, $N_{\text{ов}}$ — потомки, полученные в результате применения операторов: кроссинговера, мутации, инверсии, транслокации, транспозиции, сегрегации, удаления, вставки.

Отметим, что оператор редукции может применяться после каждого оператора или после всех в одной генерации генетического алгоритма. Выделяют две основных схемы редукции (иногда их называют схемы отбора).

1. *Элитная схема редукции*. В группу удаления из популяции включаются такие хромосомы, как $P^t_{k+1} \in P^t$ и только те потомки, для которых выполняется условие:

$$(\forall P^t_{k+1} \in P^t)(\exists P_{k(\text{ГО})})(\text{ЦФ}(P_{k(\text{ГО})}) > \text{ЦФ}(P^t_k)), \quad k = \overline{1,n},$$

где $P_{k(\text{ГО})}$ — потомки (решения), полученные после применения генетического оператора (ГО).

2. *Последовательная схема редукции* позволяет варьировать методы выбора хромосом для удаления из популяции:
- случайный выбор,
- выбор «лучших» и «худших»,
- «близкое» родство,
- «дальнее» родство,
- на основе кода Грея для бинарных хромосом,
- на основе расстояние Хэмминга,
- на основе «турнира».

Случайный выбор хромосом позволяет разнообразить генофонд на ранних этапах генетических алгоритмов. Вероятность этого выбора должна снижаться при эволюции поколений.

По аналогии с оператором репродукции известны следующие модификации операторов редукции:

1) равновероятностный отбор с вероятностью

$$P_k(s) = \frac{1}{N},$$

где N — размер популяции;

2) пропорциональный отбор с вероятностью

$$P_k(s) = \frac{\text{ЦФ}(P_k)}{\sum\limits_{k=1}^{n} \text{ЦФ}(P_k)}.$$

Подведем итоги. С помощью операторов редукции на ранних стадиях работы генетических алгоритмов происходит выбор хромосом без

учета значений их ЦФ(P_k), т. е. случайный отбор. На заключительной стадии — определяющий фактор при отборе значения ЦФ(P_k). Чем выше ЦФ(P_k), тем выше вероятность отбора P_k в следующую популяцию. На заключительной стадии проводится уменьшение случайных операций и увеличивается процент направленных.

5.17. Рассмотрим теперь ***оператор рекомбинации***. Оператор рекомбинации — языковая конструкция, которая определяет, как новая генерация хромосом будет построена из родителей и потомков. Другими словами, оператор рекомбинации — это технология анализа и преобразования популяции при переходе из одной генерации в другую. Существует много путей выполнения рекомбинации. Один из них состоит из перемещения родителей в потомки после реализации каждого генетического оператора. Другой путь заключается в перемещении некоторой части популяции после каждой генерации.

Часто в генетических алгоритмах задается параметр $W(P)$, который управляет этим процессом. Так, $N_p(1 - W(P))$ элементов в популяции P, выбранных случайно, могут «выжить» в следующей генерации. Здесь N_p — размер популяции. Величина $W(P) = 0$ означает, что целая предыдущая популяция перемещается в новую популяцию в каждой генерации. При дальнейшей реализации алгоритма лучшие или отобранные элементы из родителей и потомков будут выбираться для формирования новой популяции.

В инженерных задачах используются различные механизмы и модели этого процесса. Приведем несколько из них:

- M1 — вытеснение (`crowding factor`). Этот механизм определяет способ и порядок замены родительских хромосом из генерации t хромосомами потомками после генерации $t + 1$. Механизм реализован таким образом, что стремится удалять «похожие» хромосомы из популяции и оставлять отличающиеся.
- M2 — разделение (`sharing`). Этот механизм вводит зависимость значения целевой функции хромосомы от их распределения в популяции и поисковом пространстве. Это позволяет копиям родительских хромосом или близких к ним не появляться в популяциях.
- M3 — введение идентификаторов (`tagging`). Специальным хромосомам присваиваются метки. Операторы генетических алгоритмов применяются только к помеченным хромосомам.

Отметим, что оператор редукции является частным случаем оператора рекомбинации.

5.18. Важным понятием при реализации генетических операторов является вероятность, которая определяет, какой процент общего числа генов в популяции изменяется в каждой генерации. Для оптимизационных задач вероятность оператора кроссинговера обычно принимают в пределах $(0{,}6 \div 0{,}99)$; вероятность оператора мутации — 0,6; инверсии — $(0{,}1 \div 0{,}5)$; транслокации — $(0{,}1 \div 0{,}5)$; транспозиции —

(0,1 ÷ 0,5); сегрегации — (0,6 ÷ 0,99); удаления — (0,6 ÷ 0,99); вставки — (0,6 ÷ 0,99).

3.3. Теоретико-множественные операции над популяциями и хромосомами

1. Вид, популяция, хромосома, ген являются соответствием некоторых множеств альтернативных решений рассматриваемых задач или их частей. Основателем теории множеств является Г. Кантор. По его определению, ***множество*** — это любое объединение в одно целое определенных, вполне различимых объектов из нашего восприятия или мысли. В этой связи приведем известные операции над множествами, применяемые к популяциям и хромосомам в генетических алгоритмах.

1.1. ***Объединением*** популяций (хромосом) A и B будем считать популяцию (хромосому) C, которая состоит из элементов, принадлежащих или популяции (хромосоме) A, или популяции (хромосоме) B, или обеим популяциям (хромосомам) одновременно:

$$C = A \cup B,$$

где $\cup$ — знак объединения,

$$P_i \in C = A \cup B \rightarrow P_i \in A \vee P_i \in B.$$

Эта запись означает, что ***если*** популяция P_i принадлежит популяции C, ***то*** P_i принадлежит популяции A ***или*** P_i принадлежит популяции B.

Пример 1.

$A = \{P_1, P_2, P_3, P_4\}$,
$B = \{P_2, P_4, P_6, P_8\}$,
$A \cup B = C = \{P_1, P_2, P_3, P_4, P_6, P_8\}$.

Можно объединить не только две, но и любое количество популяций (хромосом):

$$A_1 \cup A_2 \cup A_3 \cup \ldots \cup A_n = \bigcup_{i=1}^{n} A_i$$

1.2. Популяция C считается ***пересечением популяций*** A и B, если популяция C состоит из элементов, которые принадлежат одновременно и популяции A, и популяции B:

$$C = A \cap B,$$

где $\cap$ — знак пересечения.

$$P_i \in C = A \cap B \rightarrow P_i \in A \wedge P_i \in B.$$

Эта запись означает, что ***если*** популяция P_i принадлежит популяции C, ***то*** P_i принадлежит популяции A ***и*** P_i принадлежит популяции B.

Пример 2.
$A = \{P_1, P_2, P_3, P_4\}$,
$B = \{P_5, P_4, P_6, P_8\}$,
$A \cap B = C = \{P_4\}$.
Пример 3.
$A_1 = \{P_1, P_2, P_3, P_4\}$,
$B_1 = \{P_5, P_6, P_8\}$,
$C = A \cap B = \varnothing$.

Операция пересечения может выполняться над любым количеством видов, популяций, хромосом и генов.

1.3. Популяция C, равная $A \backslash B$, называется ***разностью популяций*** A и B, если C состоит из элементов, которые принадлежат A и не принадлежат B:

$$C = A \backslash B,$$

где $\backslash$ — знак разности,

$$P_i \in C = A \backslash B \rightarrow P_i \in A \wedge P_i \notin B.$$

Пример 4.
$A = \{P_1, P_2, P_3, P_4\}$,
$B = \{P_2, P_4, P_6, P_8\}$,
$A \backslash B = C = \{P_1, P_3\}$,
$B \backslash A = D = \{P_6, P_8\}$.
Пример 5.
$A_1 = \{P_1, P_2, P_3, P_4\}$,
$B_1 = \{P_5, P_6, P_8\}$,
$C_1 = A_1 \backslash B_1 = \{P_1, P_2, P_3, P_4\}$.

В отличие от объединения и пересечения операция разности применяется только для двух популяций или их элементов.

2. Ранее рассмотренные множества популяций состояли из неупорядоченных элементов. Отметим, что последовательность элементов в популяции может быть строго задана. В этом случае каждая хромосома популяции называется кортежем. ***Кортеж*** — это упорядоченное множество. Синонимами термину кортеж являются вектор и набор.

$\alpha = \langle 3, 2, 5, 4 \rangle$ — пример упорядоченной хромосомы (кортежа).

Кортеж α состоит из четырех компонент (генов). Ген «3» расположен на первом месте (в первом локусе), «2» — на втором месте (во втором локусе), «5» — на третьем, «4» — на четвертом месте. В отличие от множества кортеж может иметь повторяющиеся элементы.

Число компонент кортежа (хромосомы) называется его ***длиной***. Кортежем длины 2 называется двойка (пара), длины 3 — тройка и так далее. Кортеж, который не содержит элементов в своем составе, называется пустым кортежем, обозначается $\langle\ \rangle$. Кортеж обозначается строчными греческими буквами. Компоненты кортежа — строчными латинскими буквами. Говорят, что два кортежа равны ($\alpha = \beta$), если

α и β имеют одинаковую длину и каждая компонента кортежа α совпадает с каждой компонентой кортежа β. Например,

$$\langle a, b\rangle = \langle c, d\rangle \leftrightarrow a = c \wedge b = d,$$
$$\langle a1, a2, a3\rangle = \langle b1, b2, b3\rangle \leftrightarrow a1 = b1 \wedge a2 = b2 \wedge a3 = b3.$$

Кортежи α и β неравны ($\alpha \neq \beta$), если имеют разную длину и/или отдельные компоненты кортежа α не совпадают с отдельными элементами кортежа β.

В отличие от множеств порядок следования компонент кортежа существенен. Отметим, что иногда кортеж называют конечным упорядоченным множеством.

2.1. ***Прямым*** или ***декартовым произведением популяций*** A и B называется популяция, состоящая из всех тех и только тех пар, т. е. кортежей длины 2, первая компонента которых принадлежит популяции A, а вторая — популяции B:

$A \times B$ — прямое декартово произведение популяций A и B;
$A = \{P_1, P_2\}$,
$B = \{P_3, P_4\}$,
$A \times B = \{\langle P_1, P_3\rangle, \langle P_1, P_4\rangle, \langle P_2, P_3\rangle, \langle P_2, P_4\rangle\}$.

Прямое произведение можно построить не только для двух, но и для трех, четырех и более популяций.

Легко видеть, что декартово произведение $A \times B$ равно $\varnothing$ (пусто), если

$$A = \varnothing \vee B = \varnothing, \;\; A \times B = \varnothing \rightarrow A = \varnothing \vee B = \varnothing.$$

2.2. Инверсия кортежа определяется через инверсию его элементов. Кортеж $\langle P_3, P_1\rangle$ называется ***инверсией*** кортежа $\langle P_1, P_3\rangle$.

Инверсия кортежа α обозначается α^{-1}:
$\alpha = \langle P_1, P_3\rangle$, $\alpha^{-1} = \langle P_3, P_1\rangle$.
$(\alpha^{-1})^{-1} = \alpha$.

2.3. Введем понятие операции композиции двух хромосом. Хромосома R называется ***композицией*** двух хромосом M и Q тогда и только тогда, когда существует элемент P_i, такой, что $P_i \in M$ и является последним в M, и $P_i \in Q$ и является первым в Q. Например,

$$\langle P_2, P_1, P_4\rangle \in M \wedge \langle P_4, P_5, P_3\rangle \in Q \rightarrow \langle P_2, P_1, P_5, P_3\rangle \in R.$$

3. ***Отношение*** — связь между любыми объектами в природе. ***Отношение между популяциями*** — это пара, причем упорядоченная, первая компонента которой является новой популяцией, а вторая — областью задания отношения. В отличие от понятия множества понятие отношения является определенным понятием.

Выражение $\varphi = \langle \Phi, M\rangle$ называется отношением, если $\Phi \subseteq M^2$, т. е. $\Phi \subseteq M \times M$, где φ называется отношением; Φ — графиком отношения

(например, популяция альтернативных решений практических задач); M — областью задания отношения; M представляет собой неупорядоченное множество, например, область работоспособности.

Отношение такого типа называется бинарным. Если $\Phi \subseteq M^k$, то отношение называется k-арным. Отношение φ с областью задания M называется отношением на множестве M.

Например, пусть мы имеем две популяции A и B, причем $\langle A, B\rangle \in \in \Phi$. Тогда может существовать отношение $A\,\varphi\,B$, в котором элементы A и B находятся в некоторой связи (например, A и B — альтернативные решения одной и той же задачи), или элемент A находится в отношении φ к элементу B.

Рассмотрим пример отношений:

$M = \{1, 2, 3, 4\}, \Phi = \{\langle 1, 1\rangle, \langle 1, 2\rangle, \langle 2, 2\rangle, \langle 3, 2\rangle\}$.

$M^2 = M \times M = \{\langle 1, 1\rangle, \langle 1, 2\rangle, \langle 1, 3\rangle, \langle 1, 4\rangle, \langle 2, 1\rangle, \langle 2, 2\rangle, \ldots, \langle 4, 4\rangle\}$.

Существует способ задания отношений через высказывательные формы:

$$x\,\varphi\,y \leftrightarrow x < y \text{ — это есть отношение } < \text{ (меньше) между элементами } x \text{ и } y.$$

Тогда $5\,\varphi\,3$ является отношением ложным, а $3\,\varphi\,4$ является отношением истинным.

3.2. Приведем основные свойства отношений.

Отношение $\varphi = \langle \Phi, M\rangle$ называется *полным отношением*, если $\Phi = M^2$, т. е. $(\forall x \in M)(\forall y \in M)(x\,\varphi\,y)$ всегда истинно, т. е. всегда существует взаимосвязь между двумя любыми популяциями x, y в области решений M.

Отношение $\varphi = \langle \Phi, M\rangle$ называется *отношением равенства*, если $\Phi = \Delta M$. Или $(\forall x, y \in V)(x\,\varphi\,y \to x = y)$. Другими словами, для любых двух популяций из множества решений M истинно высказывание: «если популяция x находится в отношении φ к популяции y, то эти популяции равны». Например, если

$$M = \{1, 2, 3\}, \quad \Phi = \{\langle 1, 1\rangle, \langle 2, 2\rangle, \langle 3, 3\rangle\},$$

то $\varphi = \langle \Phi, M\rangle$ является отношением равенства.

Отношение $\varphi = \langle \Phi, M\rangle$ называется *отношением неравенства*, если $\Phi = M^2 \backslash \Delta M$, если $(\forall x, y \in M)(x\,\varphi\,y \to x \neq y)$. Другими словами, для любых двух популяций из множества решений M истинно высказывание: «если популяция x находится в отношении φ к популяции y, то эти популяции неравны».

3.3. На отношения между видами, популяциями, хромосомами и генами переносятся основные операции над множествами, но они могут выполняться только на одной и той же области задания.

Объединением отношений популяций φ_1 и φ_2 на множестве решений M называется отношение φ_3.

$\varphi_1 \cup \varphi_2 = \varphi_3$, $\varphi_1 = \langle \Phi_1, M\rangle$, $\varphi_2 = \langle \Phi_2, M\rangle$.

$\varphi_3 = \langle \Phi_1 \cup \Phi_2, M\rangle$.

Пример 6.
$\varphi_1 = \langle \Phi_1, M\rangle$, $\varphi_2 = \langle \Phi_2, M\rangle$,
$M = \{2,3,4\}$, $\Phi_1 = \{\langle 2,1\rangle, \langle 2,2\rangle, \langle 2,4\rangle\}$,
$\Phi_2 = \{\langle 2,1\rangle, \langle 2,3\rangle, \langle 4,4\rangle\}$,
$\Phi_3 = \Phi_1 \cup \Phi_2 = \{\langle 2,1\rangle, \langle 2,2\rangle, \langle 2,4\rangle, \langle 2,3\rangle, \langle 4,4\rangle\}$.
Отметим следующее, очевидно, что

$$x(\varphi_1 \cup \varphi_2)y \leftrightarrow x\,\varphi_1\, y \vee x\varphi_2 y.$$

Другими словами, если популяции x, y находятся в отношении $\varphi_1 \cup \varphi_2$, то это эквивалентно высказыванию: популяция x находится в отношении φ_1 к популяции y или популяция x находится в отношении φ_2 к популяции y.

Пересечением отношений φ_1 и φ_2 на множестве M называется отношение φ_3:
$\varphi_1 \cap \varphi_2 = \varphi_3$, $\varphi_1 = \langle \Phi_1, \mathrm{M}\rangle$, $\varphi_2 = \langle \Phi_2, \mathrm{M}\rangle$.
$\varphi_3 = \langle \Phi_1 \cap \Phi_2, \mathrm{M}\rangle$.
Пример 7.
$M = \{1,2\}$, $\varphi_1 = \langle\{\langle 1,1\rangle, \langle 1,2\rangle\}, \{1,2\}\rangle$,
$\varphi_2 = \langle\{\langle 1,2\rangle, \langle 2,2\rangle\}, \{1,2\}\rangle$,
$\varphi_1 \cap \varphi_2 = \langle\{\langle 1,2\rangle\}, \{1,2\}\rangle$.

Отношение φ_3 называется *разностью отношений* φ_1 и φ_2, если
$\varphi_3 = \langle \Phi_1 \backslash \Phi_2, M\rangle$,
$\varphi_3 = \varphi_1 \backslash \varphi_2$.
Пример 8.
$M = \{1,2,3\}$, $\Phi_1 = \{\langle 1,1\rangle, \langle 2,2\rangle, \langle 3,3\rangle\}$,
$\Phi_2 = \{\langle 1,1\rangle, \langle 2,1\rangle, \langle 3,2\rangle\}$,
$\Phi_3 = \Phi_1 \backslash \Phi_2 = \{\langle 2,2\rangle, \langle 3,3\rangle\}$,
$\varphi_3 = \langle\{\langle 2,2\rangle, \langle 3,3\rangle\}, \{1,2,3\}\rangle$.

Отметим, что все операции над отношениями должны выполняться на одной и той же области задания, и в результате выполнения операций снова получается отношение с той же самой областью задания.

3.4. Продолжим знакомство с основными свойствами отношений.

Отношение $\varphi = \langle \Phi, M\rangle$ называется отношением *рефлексивности*, если $(\forall x \in M)(x\,\varphi\, x)$.

Отношение рефлексивности является унарной операцией, т. е. операцией над одним элементом. Отношение равенства (параллельности) является примером отношения рефлексивности. Отношение перпендикулярности ($x \perp y$) не является отношением рефлексивности.

Отношение $\varphi = \langle \Phi, M\rangle$ называют отношением *симметричности*, если $(\forall x, y \in M)(x\,\varphi\, y \to y\,\varphi\, x)$. Другими словами, для любых двух популяций из множества решений M истинно высказывание: «если популяция x находится в отношении φ к популяции y, то и популяция y находится в отношении φ к популяции x». Отношение симметричности является бинарной операцией.

Отношение $\varphi = \langle \Phi, M \rangle$ называется отношением *транзитивности*, если $(\forall x, y, z \in M)(x \varphi y \& y \varphi z \to x \varphi z)$. Другими словами, для любых трех популяций из множества решений M истинно высказывание: «если популяция х находится в отношении φ к популяции y и популяция y находится в отношении φ к популяции z, то это означает, что популяция x находится в отношении φ к популяции z».

Пример 9.

Полное отношение $\langle M^2, M \rangle$ рефлексивно, транзитивно:

$$M = \{1, 2\}, \quad M^2 = \{\langle 1, 2\rangle, \langle 2, 1\rangle, \langle 1, 1\rangle, \langle 2, 2\rangle\}.$$

Пустое отношение является носителем всех свойств.

Пример 10.

Интерес представляют отношения, которые обладают комбинациями некоторых свойств.

Отношение φ называется отношением *эквивалентности*, если оно рефлексивно, симметрично и транзитивно:

$$\varphi \sim \varphi, \text{ где } \sim \text{ есть знак эквивалентности.}$$

Например:

равенство и параллельность — отношения эквивалентности.

Отношение порядка связано с отношениями типа $\leqslant, <, >, \geqslant$.

3.5. В тех случаях, когда x означает множество людей или группы людей, приходится сталкиваться с отношением, которое является отношением доминирования.

Будем говорить, что хромосома x доминирует над хромосомой y и писать $x \gg y$, если x в чем-то превосходит y. Так, значение целевой функции хромосомы x может превосходить значение целевой функции хромосомы y. Будем говорить, что между элементами множества x имеет место отношение доминирования, если эти элементы обладают следующими двумя свойствами:

1) никакой элемент не может доминировать самого себя, т. е. высказывание $x \gg x$ — ложно;

2) в каждой паре элементов в точности один элемент доминирует второго, т. е. высказывание $x \gg y$ и $y \gg x$ — исключается.

Говорят, что между множествами (популяциями или хромосомами) X, Y установлено *взаимнооднозначное соответствие*, если указано какое-то подмножество $G \subseteq X \times Y$, которое обладает некоторыми свойствами. Тогда соответствие (Γ) — это тройка множеств $\Gamma = \langle G, X, Y \rangle$, первая компонента которой является графиком G, вторая компонента является множеством X и третья — множеством Y:

$$\Gamma = \langle G, X, Y \rangle, \quad G \subseteq X \times Y \text{ — определение соответствия.}$$

Другими словами, если каждому элементу популяции X соответствует один и только один элемент популяции Y, то такое соответствие является взаимнооднозначным.

Множество X называется областью отправления соответствия, а множество Y — областью прибытия соответствия:

$$G \subseteq X \times Y \leftrightarrow G = X \times Y \vee G \subset X \times Y.$$

Если кортеж из двух хромосом $\langle a, b\rangle \in G$, то можно сказать, что хромосома (b) соответствует хромосоме (a) в соответствии Γ. Приведем ряд примеров.

1) $\Gamma = \langle\{\langle a, \alpha\rangle, \langle b, \beta\rangle, \langle c, \alpha\rangle\},\ X = \{a, b, c\},\ Y = \{\alpha, \beta\}\rangle$.
Если $X = Y$, то соответствие Γ превращается в отношение.

Можно сказать, что отношение является частным случаем соответствия. Все свойства отношений можно переносить на соответствия.

2) $\Gamma = \langle G, X, Y\rangle,\ X = \{1, 2, 3\},\ Y = \{4, 5\}, G = X \times Y$.

3) Пусть имеем два множества (популяции): $X = \{b, c\}$, $Y = \{\alpha, \beta\}$, тогда $\Gamma = \langle\{\langle b, \alpha\rangle, \langle c, \beta\rangle\},\ \ X = \{b, c\},\ Y = \{\alpha, \beta\}\}\rangle$ является примером взаимнооднозначного соответствия между популяциями, так как каждому элементу X соответствует один и только один элемент Y.

4. Выше рассматривались четкие множества. В них можно однозначно сказать, является ли рассматриваемый объект элементом этого множества. Следовательно, эти множества имеют четкие границы, определяющие принадлежность или не принадлежность элемента множеству. В генетических алгоритмах не всегда можно провести четкую грань между принадлежностью и не принадлежностью элементов множеству.

4.1. ***Нечеткое*** (***расплывчатое***) ***множество***, согласно Л. Заде, — это множество, которое имеет нечеткие границы, определяющие принадлежность или не принадлежность элемента множеству.

Пусть задана произвольная непустая упорядоченная последовательность (хромосома): $P_i = \langle p_1, p_2, \ldots, p_n\rangle$.

В данной хромосоме каждый элемент p_i есть кортеж длины два $\langle\mu_{\sim A}(x_i), x_i\rangle$, первая компонента которого обозначается $\mu_{\sim A}(x_i)$ и называется числовой функцией, она задается на сегменте $[0, 1]$. Второй элемент кортежа — элемент x_i множества X ($x_i \in X$) — множество ограничений или множество параметров рассматриваемой оптимизационной задачи.

При задании некоторого решения хромосомы P_i каждому элементу x_i присваивается число $0 \leqslant \mu_{\sim A}(x_i) \leqslant 1$, это число и определяет степень влияния («важности») данного параметра или, иначе говоря, его весовой коэффициент для данного решения P_i. В частном случае величина $\mu_{\sim A}(x_i)$ является значением целевой функции для хромосомы P_i.

Например, имеется некая оптимизационная задача: $Q \to \max(\min)$, имеющая n ограничений $\sum\limits_{i=1}^{n} p_i \geqslant 0$. Тогда можно сформировать множество P допустимых решений данной задачи (хромосом). Любое реше-

ние P_i из множества P можно представить, например, в следующем виде:

$$P_i = \{\langle 0{,}1, x_1\rangle, \langle 1, x_2\rangle, \langle 0, x_3\rangle, \langle 0{,}5, x_4\rangle, \ldots, \langle 0{,}8, x_n\rangle\}.$$

Весовая функция каждого параметра решения может задаваться субъективно, в частности, на основе экспертных оценок. Подобным образом можно представить также все множество решений (популяцию):

$$P = \{\langle \mu_{\sim A}(P_i), P_i\rangle\}, \text{ где } \mu_{\sim A}(P_i) \text{ — целевая функция решения } P_i.$$

4.2. Рассмотрим понятие нечеткой алгебры для решений, имеющих числовые коэффициенты. При помощи предлагаемых правил образуются новые решения из уже имеющихся с помощью нечетких логических операций: отрицания, дизъюнкции, конъюнкции, импликации и эквивалентности.

Отрицанием решения P_i называется решение, числовой коэффициент которого определяется по формуле

$$\overline{A}_i = 1 - A_i.$$

Конъюнкцией двух решений P_i, P_j называется решение, числовой коэффициент которого определяется по формуле

$$P_i \& P_j = \min(P_i, P_j).$$

Дизъюнкцией двух решений P_i, P_j называется решение, числовой коэффициент которого определяется по формуле

$$P_i \vee P_j = \max(P_i, P_j).$$

Импликацией двух решений P_i, P_j называется решение, числовой коэффициент которого определяется по формуле

$$P_i \to P_j = \max(1 - P_i, P_j).$$

Эквивалентностью двух решений P_i, P_j называется решение, числовой коэффициент которого определяется по формуле

$$P_i \leftrightarrow P_j = \min\big(\max(1 - P_i, P_j);\ \max(P_i, 1 - P_j)\big).$$

Данные правила справедливы при условии, что числовые коэффициенты решений находятся в интервале $[0, 1]$.

Пример 11. Пусть $P_1 = 0{,}7$, $P_2 = 0{,}4$. Тогда:

$P_1 \wedge P_2 = \min(P_1, P_2) = 0{,}4;$

$P_1 \vee P_2 = \max(P_1, P_2) = 0{,}7;$

$P_1 \to P_2 = \max(1 - P_1, P_2) = \max(1 - 0{,}7; 0{,}4) = 0{,}4;$

$P_1 \leftrightarrow P_2 = \min(\max(1 - P_1, P_2);\ \max(P_1, 1 - P_2)) =$

$= \min(\max(1 - 0{,}7; 0{,}4);\ \max(0{,}7; 1 - 0{,}4)) = \min(0{,}4; 0{,}7) = 0{,}4.$

5.3. Рассмотрим операции над расплывчатыми популяциями. Пусть заданы расплывчатые популяции $\sim A$, $\sim B$, X. Введем степень включения $\nu(\sim A, \sim B)$ для $\sim A$, $\sim B$. Она определяется по формуле

$$\nu(\sim A, \sim B) = \wedge(\mu_{\sim A}(x) \to \mu_{\sim B}(x)), \quad x \in X.$$

Если $\nu(\sim A, \sim B) \geqslant 0{,}5$, то считается, что расплывчатая популяция $\sim A$ нестрого включается в $\sim B$: $\sim A \subseteq \sim B$.

Если $\nu(\sim A, \sim B) < 0{,}5$, то $\sim A$ не включается в $\sim B$.

Пример 12.

$X = \{x_1, x_2, x_3, x_4, x_5\}$,
$\sim A = \{\langle 0{,}3; x_2\rangle, \langle 0{,}7; x_3\rangle, \langle 0{,}9; x_4\rangle\}$,
$\sim B = \{\langle 0{,}1; x_1\rangle, \langle 0{,}8; x_2\rangle, \langle 0{,}3; x_3\rangle, \langle 0{,}9; x_5\rangle\}$,
$\nu(\sim A, \sim B) = (0 \to 0{,}1) \wedge (0{,}3 \to 0{,}8) \wedge (0{,}7 \to 0{,}3) \wedge$
$\wedge(0{,}9 \to 0) \wedge (0 \to 0{,}9) = 1 \wedge 0{,}7 \wedge 0{,}7 \wedge 1 \wedge 1 = 0{,}7.$

Следовательно, $\sim A \subseteq \sim B$. Отметим, что степень включения одной расплывчатой популяции в другую может быть определена для любых двух расплывчатых множеств, при этом она может принимать любое значение от 0 до 1.

Рассмотрим основные операции над расплывчатыми популяциями.

1) Дано $X, \sim A, \sim B$. Пусть

$$\sim A = \{\langle \mu_{\sim A}(x), x\rangle, x \in X\}, \quad \sim B = \{\langle \mu_{\sim B}(x), x\rangle, x \in X\}.$$

Объединением расплывчатых популяций называется выражение вида

$$\sim A \cup \sim B = \{\langle \mu_{\sim A \cup \sim B}(x), x\rangle, x \in X\}.$$

Значение $\mu_{\sim A \cup \sim B}$ находится согласно следующей формуле:

$$\mu_{\sim A \cup \sim B} = \mu_{\sim A} \vee \mu_{\sim B},$$

Пример 13. Пусть
$X = \{x_1, x_2, x_3, x_4\}$,
$\sim A = \{\langle 0{,}1; x_1\rangle, \langle 1; x_3\rangle\}$,
$\sim B = \{\langle 0{,}8; x_1\rangle, \langle 1; x_2\rangle, \langle 0{,}3; x_3\rangle, \langle 0{,}7; x_4\rangle\}$,
$\sim A \cup \sim B = \{\langle 0{,}8; x_1\rangle, \langle 1; x_2\rangle, \langle 1; x_3\rangle, \langle 0{,}7; x_4\rangle\}$.

2) $\sim A \cap \sim B$ — пересечение расплывчатых популяций, если

$$\sim A \cap \sim B = \{\langle \mu_{\sim A \cap \sim B}(x), x\rangle, x \in X\},$$

где

$$\mu_{\sim A \cap \sim B} = \mu_{\sim A} \wedge \mu_{\sim B}.$$

Пример 14. Условие см. выше.

$$\sim A \cap \sim B = \{\langle 0{,}1; x_1\rangle, \langle 0; x_2\rangle, \langle 0{,}3; x_3\rangle, \langle 0; x_4\rangle\}.$$

3) $\sim A\setminus \sim B$ называется *разностью расплывчатых популяций*, если

$$\sim A\setminus \sim B = \{\langle \mu_{\sim A\setminus \sim B}(x), x\rangle, x \in X\},$$

где

$$\mu_{\sim A\setminus \sim B} = \mu_{\sim A} \wedge \neg \mu_{\sim B}.$$

Величина $\mu_{\sim A\setminus \sim B}$ определяется согласно вышеприведенным формулам.

Пример 15. Условие см. выше.

$\sim A\setminus \sim B = \{\langle 0{,}1; x_1\rangle, \langle 0; x_2\rangle, \langle 0{,}7; x_3\rangle, \langle 0; x_4\rangle\}$.

Пример 16.

$\sim A = \{\langle 0{,}1; x_1\rangle, \langle 0{,}7; x_2\rangle\},\ \sim B = \{\langle 0{,}4; x_1\rangle, \langle 0{,}3; x_2\rangle\}$.

а) $\mu_{\sim A\setminus \sim B}(x) = \mu_{\sim A}(x) \wedge \neg \mu_{\sim B}(x),\ \mu_{\sim A\setminus \sim B}(x) = \{\langle 0{,}1; x_1\rangle, \langle 0{,}7; x_2\rangle\}$.

б) $\mu_{\sim B\setminus \sim A}(x) = \mu_{\sim B}(x) \wedge \neg \mu_{\sim A}(x),\ \neg \mu_{\sim A}(x) = \{\langle 0{,}9; x_1\rangle, \langle 0{,}3; x_2\rangle\},$
$\mu_{\sim B\setminus \sim A}(x) = \{\langle 0{,}4; x_1\rangle, \langle 0{,}3; x_2\rangle\}$.

Описанные операции над четкими и расплывчатыми множествами эффективно применяются при построении генетических операторов и алгоритмов.

3.4. Простой генетический алгоритм

1. Эволюционный процесс представляется как способность «лучших» хромосом оказывать большее влияние на состав новой популяции на основе длительного выживания из более многочисленного потомства. Основные этапы эволюционного поиска следующие.

1. Сконструировать начальную популяцию. Ввести точку отсчета поколений $t = 0$. Вычислить приспособленность каждой хромосомы в популяции, а затем среднюю приспособленность всей популяции.
2. Установить $t = t + 1$. Произвести выбор двух родителей (хромосом) для реализации оператора кроссинговера. Он выполняется случайным образом пропорционально приспособляемости родителей.
3. Сформировать генотип потомков. Для этого с заданной вероятностью произвести оператор кроссинговера над генотипами выбранных хромосом. Далее с вероятностью 0,5 выбрать один из потомков $P_i(t)$ и сохранить как член новой популяции. После этого к $P_i(t)$ последовательно применить оператор инверсии, а затем — оператор мутации с заданными вероятностями. Полученный генотип потомка сохранить как $P_k(t)$.
4. Определить количество хромосом для исключения их из популяции, чтобы ее размер оставался постоянным. Текущую популяцию обновить заменой отобранных хромосом на потомков $P_k(t)$.
5. Произвести определение приспособленности (целевой функции) и пересчет средней приспособленности всей полученной популяции.
6. Если $t = t_{\text{заданному}}$, то перейти к 7, если нет, то перейти к 2.
7. Конец работы.

Данный алгоритм известен как упрощенный «репродуктивный план Д. Холланда». Заметим, что в практических задачах вместо понятия «приспособленность» используют понятие «целевая функция».

2. ***Простой генетический алгоритм*** был впервые описан Д. Гольдбергом на основе работ Д. Холланда. Его механизм несложен. Предварительно простой генетический алгоритм случайно генерирует популяцию последовательностей — хромосом (альтернативных упорядоченных и неупорядоченных решений). Затем производится копирование последовательности хромосом и перестановка их частей. Далее простой генетический алгоритм реализует множество простых операций к начальной популяции и генерирует новые решения.

Простой генетический алгоритм состоит из трех операторов:

- репродукции;
- кроссинговера;
- мутации.

Репродукция — процесс, в котором хромосомы копируются пропорционально значению их целевой функции. Копирование хромосом с «лучшим» значением целевой функции имеет бóльшую вероятность для попадания в следующую генерацию. Рассматривая эволюцию Дарвина, можно отметить, что оператор репродукции является искусственной версией натуральной селекции — «выживание сильнейших». Он представляется в алгоритмической форме различными способами. Самый простой — создать модель «колеса рулетки», в которой каждая хромосома имеет поле, пропорциональное значению целевой функции.

3. Рассмотрим пример Д. Гольдберга: необходимо найти значение максимума функции $f(x) = x^2$ на целочисленном интервале $[0, 31]$. Традиционными методами можно изменять значения переменной x, пока не получим максимальное значение $f(x)$.

Для объяснения и реализации простого генетического алгоритма построим следующую таблицу (табл. 3.1). В столбце 2 табл. 3.1 расположены 4 хромосомы (представленные в двоичном коде), сгенерированные случайным образом. Значение целевой функции для каждой хромосомы (столбец 4) определяется как квадрат значения десятичного кода двоичного числа, которое приведено для хромосом во втором столбце таблицы. Тогда суммарное значение целевой функции всех хромосом равно 890. Для селекции хромосом используется оператор репродукции на основе колеса рулетки.

На рис. 3.2 поля колеса рулетки соответствуют значению целевой функции каждой хромосомы в процентах. В одной генерации колесо рулетки вращается, и после остановки ее указатель определяет хромосому, выбранную для реализации следующего оператора. Очевидно, не всегда хромосома с большим значением целевой функции в результате применения оператора репродукции будет выбрана для дальнейших преобразований.

Таблица 3.1

Номер хромосом	Хромосома		Значение ЦФ	Значение ЦФ, в процентах
	(двоичный код)	Десятичный код		
1	0 1 1 0 0	12	144	16,2
2	1 0 0 0 0	16	256	28,8
3	0 0 1 1 1	7	49	5,5
4	1 0 1 0 1	21	441	49,5

Для упрощения в рассматриваемом примере будем считать, что $16{,}2 = 16$; $49{,}5 = 50$; $5{,}5 = 5$; $28{,}8 = 29$ (рис. 3.2).

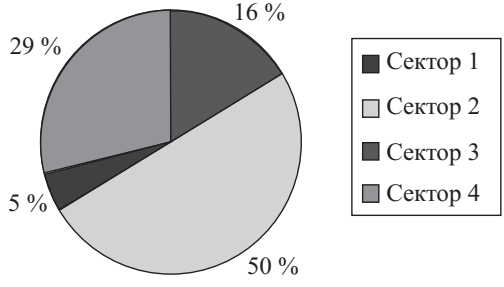


Рис. 3.2. Колесо рулетки для примера

На основе реализации оператора репродукции выбираются хромосомы для применения оператора кроссинговера. Оператор кроссинговера, как правило, выполняет в 3 шага, одним из операторов кроссинговера, описанным выше. Точка разрыва k выбирается случайно между 1 и числом, равным длине хромосомы минус единица, т. е. в интервале $(1, L-1)$. Длина хромосомы L — это число значащих цифр в ее коде. В рассматриваемом примере (табл. 3.1) длина каждой хромосомы равна пяти ($L = 5$). На основе оператора кроссинговера создаются две новые хромосомы путем обмена их частей между позициями $(k+1)$ и L соответственно.

Например, рассмотрим хромосомы 1 и 2 из начальной популяции (табл. 3.1). Пусть $k = 1$. Тогда получим:

P_1 : 0 | 1 1 0 0 — до применения оператора кроссинговера,
P_2 : 1 | 0 0 0 0

P_1' : 0 0 0 0 0 — после применения оператора кроссинговера,
P_2' : 1 1 1 0 0

Работа простого генетического алгоритма начинается с репродукции. Хромосомы для следующей генерации выбираются путем враще-

ния колеса рулетки. В примере колесо рулетки вращается 4 раза. Это число соответствует мощности начальной популяции.

Величину отношения $f_i(x)/\mathrm{sum}\, f(x)$ называют вероятностью выбора копий (хромосом) при реализации оператора

$$P_i(\mathrm{OP}) = \frac{f_i(x)}{\mathrm{sum}\, f(x)},$$

где $f_i(x)$ — значение целевой функции i-й хромосомы в популяции, $\mathrm{sum}\, f(x)$ — суммарное значение целевой функции всех хромосом в популяции, OP — оператор репродукции. Величину $P_i(\mathrm{OP})$ также называют нормализованной вероятностью выбора. Ожидаемое число копий i-й хромосомы после реализации оператора репродукции определяется по формуле

$$N_i = P_i(\mathrm{OP}) \cdot N_G,$$

где N_G — число анализируемых хромосом, причем N_G включается в N.

Ожидаемое число копий хромосомы P_i, переходящее в следующее поколение, также иногда определяют на основе выражения

$$N_i = \frac{f_i(x)}{\overline{f}(x)},$$

где $\overline{f}(x)$ — среднее значение целевой функции по всей популяции.

Тогда для рассматриваемого примера ожидаемое число копий для первой хромосомы из табл. 3.2 равно $0{,}16 \cdot 4 = 0{,}64$ копий, для второй — $0{,}29 \cdot 4 = 1{,}16$ копий, для третьей — $0{,}05 \cdot 4 = 0{,}2$ и, наконец, для четвертой — $0{,}5 \cdot 4 = 2$. Используя модель «бросание монеты», можно определить число полученных копий. Например, (см. столбец 7 табл. 3.2) хромосомы P_1 и P_2 получают 1 копию, хромосома P_4 — 2 копии и хромосома 3 не получает копий. Сравнивая этот результат с ожидаемым числом копий, получаем, что «лучшие» хромосомы дают большее число копий, «средние» остаются и «плохие» удаляются после реализации оператора репродукции.

Для рассматриваемого примера построим табл. 3.3. В столбце 2 приведен вид четырех хромосом после выполнения оператора репродукции. В столбце 3 приведены списки пар хромосом, которые выбраны случайным образом для реализации оператора кроссинговера. В столбце 4 указан номер позиции для точки разреза хромосом. В столбце 5 приведен вид 4 хромосом после выполнения оператора кроссинговера. В столбце 6 приведены значения десятичного кода двоичного числа каждой хромосомы столбца 5. В столбце 7 приведено значение $f(x)$ для каждой из 4 хромосом новой популяции. В строке 5 приведено суммарное значение ЦФ (целевой функции) хромосом новой популяции, в строке 6 — среднее значение их целевой функции, а в строке 7 — максимальное значение целевой функции хромосомы из новой популяции.

Таблица 3.2

№	Начальная популяция	x	$f(x)$	Значение $f_i(x)/\text{sum}\,[f(x)]$	Ожидаемое число копий	Число полученных копий
1	0 1 1 0 0	12	144	0,16	0,65	1
2	1 0 0 0 0	16	256	0,29	1,15	1
3	0 0 1 1 1	7	49	0,05	0,22	0
4	1 0 1 0 1	21	441	0,55	1,98	2
Суммарное значение ЦФ (sum $f(x)$)			890	1,00	4,00	4
Среднее значение ЦФ $\overline{f}(x)$			222,5	0,22	0,88	1
Max значение ЦФ			441	0,5	2	2

Таблица 3.3

№	Популяция после оператора репродукции	Пары, выбранные случайно	Точка оператора кроссинговера	Новая популяция	x	$f(x)$
1	0 | 1 1 0 0	1–4	1	0 0 1 0 1	5	25
2	1 0 | 0 0 0	2–3	2	1 0 1 1 1	23	529
3	0 0 | 1 1 1	2–3	2	0 0 0 0 0	0	0
4	1 | 0 1 0 1	1–4	1	1 1 1 0 0	28	784
Sum $f(x)$	1338					
$\overline{f}(x)$	334,5					
Max $f(x)$	784					

Применяя к популяции, полученной после реализации оператора репродукции (столбец 2 табл. 3.3), оператор кроссинговера, получим новую популяцию хромосом (5-й столбец таблицы 3.3). В принципе оператор кроссинговера можно применять любое число раз. После проведения одной генерации простой генетический алгоритм улучшились все показатели: среднее и максимальное значение целевой функции.

Далее, согласно схеме выполнения простой генетический алгоритм, реализуется оператор мутации. Существует большое количество видов операторов мутации. Заметим, что эти операторы соответствуют перестановкам элементов внутри заданного множества. Очевидно, что при небольшой длине хромосомы L (порядка 10–20) можно выполнить полный перебор за приемлемое время и найти наилучшие решения.

При увеличении L до 50–200 и выше полный перебор произвести затруднительно и необходимы другие механизмы поиска. Здесь как раз и приходит на помощь направленно-случайный поиск, который реализуется на основе простой генетический алгоритм.

Применим, например, оператор мутации к хромосоме P_2 с ЦФ = = 23 (столбец 5 табл. 2.3). Выберем позиции 2, 3 и после оператора мутации получим: $P_2 : 10|111 \Rightarrow P_2' : 11011$. Снова, применяя оператором мутации к P_2' между позициями 3 и 4, получим: $P_2' : 110|11 \Rightarrow P_2'' : 11101$. Как видно, хромосома P_2'' имеет значение ЦФ = 29. Опять применим к хромосоме P_2'' оператор мутации. Тогда при случайном выборе точки оператора мутации между 4 и 5 получим: $P_2'' : 1110|1 \Rightarrow P_2''' : 11110$. Хромосома P_2''' имеет значение ЦФ = 30. Итак, найдено квазиоптимальное значение x, равное 30 для решения задачи поиска максимального значения функции $f(x) = x^2$ на интервале $[0, 31]$. Заметим, что для хромосомы P_2 лучшего значения целевой функции, чем для P_2''', найти невозможно.

Отметим, что глобальный максимум можно было найти еще на этапе реализации кроссинговера. Для этого необходимо было увеличить пространство поиска. Например, если в столбце 5 табл. 3.3 выбрать хромосомы P_2 и P_4 и между ними выполнить оператор кроссинговера (точка операторы кроссинговера выбрана случайно и равна 3), то получим:

P_2 :	1	0	1	1	1	(ЦФ-23)
P_4 :	1	1	1	0	0	(ЦФ-28)
P_2' :	1	0	1	0	0	(ЦФ-20)
P_4' :	1	1	1	1	1	(ЦФ-31)

Решение P_4', полученное в результате применения оператора кроссинговера случайным образом, является наилучшим результатом (глобальным оптимумом).

Как отмечалось выше, в генетических алгоритмах можно выделять два основных механизма воспроизводства хромосом:

- потомки являются точными копиями родителей (неполовое воспроизводство без мутации);
- потомки имеют «большие» отличия от родителей.

В генетических алгоритмах в основном используют комбинации этих механизмов. Отметим, что в инженерных задачах начальная популяция может выбираться любым образом, например, моделированием «бросания монеты» (орел = 1, решка = 0). Тогда оператор репродукции выполняется через моделирование движения колеса рулетки. Оператор кроссинговера в задачах вычислительного характера обычно выполняется через двоичное декодирование двух положений монеты. Часто применяют другие типы операторов кроссинговера и другие технологии его выполнения. Вероятность ОК (оператора кроссинговера) допускается равной $\Pr(\text{ОК}) = 1{,}0$ и меньше, вероятность ОМ (оператора мутации) допускается равной $\Pr(\text{ОМ}) = 0{,}01$ и больше. В общем слу-

чае вероятность применения оператора мутации зависит от знаний о решаемой задаче.

4. Приведем другой стандартный тип генетического алгоритма, описанный Л. Девисом:

1. Инициализировать популяции хромосом.
2. Оценить значения каждой хромосомы в популяции.
3. Создать новые хромосомы посредством скрещивания текущих хромосом; применить операторы мутации и рекомбинации.
4. Устранить хромосомы из популяции, чтобы освободить место для новых хромосом.
5. Оценить значения новых хромосом и вставить их в популяцию.
6. Если время, заданное на реализацию алгоритма, закончено, то остановиться и возвратиться к наилучшей хромосоме; если нет, то перейти к 3.
7. Конец работы алгоритма.

Сравнивая описания простых генетических алгоритмов Д. Голдберга, Д. Холланда и Л. Девиса, видим, что в них реализована одна основная идея моделирования эволюции с некоторыми модификациями. Однако заметим, что эти изменения могут существенно влиять на окончательное качество решения.

Приведем пример модифицированного простого генетического алгоритма:

1. Создать начальную популяцию решений.
2. Смоделировать популяцию (определить ЦФ (целевую функцию) для каждой хромосомы).
3. Если еще не проведено необходимое число генераций или не закончилось время, заданное на реализацию алгоритма, или не найдено оптимальное значение целевой функции (если оно известно):
   а) выбрать элементы для репродукции;
   Применить:
   б) оператор кроссинговера для создания потомков;
   в) оператор мутации;
   г) оператор инверсии;
   д) оператор транспозиции;
   е) оператор транслокации;
   ж) оператор сегрегации;
   з) оператор удаления вершин;
   и) оператор вставки вершин;
   к) рекомбинацию родителей и потомков для создания новой генерации;
   л) оператор редукции.
4. Реализовать новую генерацию.

Новые модификации генетического алгоритма могут строиться путем объединения, частичного устранения либо перестановок пунктов

«б–л», а также на основе применения адаптационных принципов управления эволюционным поиском.

3.5. Основные гипотезы генетических алгоритмов

1. В генетических алгоритмах важным понятием является «***элитизм***». Он определен как операция, когда хромосома с наилучшим значением целевой функции сразу копируется в новую популяцию. Это позволяет не терять лучшие решения в процессе эволюции.

Одним из основных вопросов в теории генетических алгоритмов является вопрос о подобии хромосом в популяции и, главным образом, о связи анализа подобия решений с эффективностью поиска.

Д. Холланд для решения этих вопросов предложил использовать алфавит, состоящий из трех символов: $\{0, 1, *\}$. Значок $*$ определяется так: «не имеет значения» и вместо него может быть использован 0 или 1. Д. Холланд ввел понятие «***шаблон***» (схема). Он описывает подмножество решений в популяции с совпадением в определенных позициях. ***Шаблон*** в генетическом алгоритме — это подмножество хромосом, содержащих знаки «*». На основании одного шаблона может быть построено некоторое подмножество дочерних хромосом, в которых знак «*» заменяется значениями 0 или 1.

Например, шаблон $(*0001)$ будет состоять из двух хромосом: (00001) и (10001). А шаблон $(*0000)$ соответствует двум хромосомам вида (10000), (00000). Другой пример: шаблон $(*111*)$ описывает множество хромосом с 4 членами (01110), (11110), (01111), (11111). Шаблон $(0 * 1 * *)$ будет уже иметь 8 хромосом длины 5.

По шаблону $(**)$ можно получить $3^2 = 9$ хромосом и шаблонов, т. е. $(**)$, $(*1)$, $(*0)$, $(1*)$, $(0*)$, (00), (01), (10), (11). В это число включаются и ***изоморфные варианты*** (***шаблоны и хромосомы***). Изоморфными называются такие варианты, из которых можно получить одинаковые хромосомы или шаблоны. Например, шаблон $(**)$ изоморфен всем остальным 8 шаблонам и хромосомам, так как из него можно получить любой из оставшихся 8 вариантов. Шаблон $(*0)$ изоморфен вариантам (00) и (10). Из этих 9 вариантов выбираются 4 ***неизоморфных*** (***хромосомы***): (00), (01), (10), (11).

Следовательно, в общем случае для шаблона длины пять ($L = 5$) можно получить $3^5 = 243$ изоморфных и неизоморфных вариантов хромосом и шаблонов. Для шаблона такой же длины ($L = 5$) получим $2^5 = 32$ различные неизоморфные хромосомы.

Например, для шаблона длины 3 вида $(* * *)$ можно получить $3^3 = 27$ изоморфных хромосом и шаблонов или $2^3 = 8$ неизоморфных хромосом (111), (110), (101), (100), (011), (010), (001), (000).

Основная идея в простом генетическом алгоритме — объединить две хромосомы со значением целевой функции выше среднего для того, чтобы получить новую хромосому с лучшим значением целевой функции, а именно найти шаблоны типа $(11 * **)$ и $(* * *111)$ при реше-

нии задач максимизации значений целевой функции. Тогда, применяя оператор кроссинговера к данным шаблонам, можно получить хромосому (11111) с наилучшим значением целевой функции при поиске максимума.

Для вычисления числа шаблонов или их границы в популяции требуются знания о каждой хромосоме в популяции. Поэтому сначала вычисляют число шаблонов, содержащихся в индивидуальной хромосоме, а затем находят верхнюю границу общего их числа в популяции.

Рассмотрим, например, хромосому длины $L = 6$, (111111). Она имеет 3^6 шаблонов, потому что каждая позиция может быть 1 или $*$. В общем случае считают, что шаблон содержит 2^L хромосом. Тогда популяция размера N_p будет содержать от 2^L до 3^L шаблонов. Шаблоны определенной короткой длины называют «строительными блоками». Размер строительных блоков очень важен для быстроты нахождения результата и его эффективности. Рассмотрим примеры строительных блоков для разных шаблонов. Например, в шаблоне $(***1)$ строительным блоком будет элемент 1. В шаблоне $(10 * **)$ строительным блоком будет составной элемент 10. Заметим, что при реализации генетического алгоритма должно выполняться условие, чтобы строительные блоки разрывались только в крайних случаях, указанных пользователем.

2. В зависимости от изменения размера популяции различают ***стационарные*** и ***поколенческие*** генетические алгоритмы. В стационарных генетических алгоритмах размер популяции является входным постоянным параметром, задаваемым пользователем. В поколенческих генетических алгоритмах размер популяции увеличивается или уменьшается в каждой последующей генерации. Следует отметить, что речь в основном идет о появлении $N_p + N_i$ потомков ($N_i \gg 1$), прежде чем начинает реализовываться оператор отбора, устраняющий N_i хромосом с худшим значением целевой функции. Вопросы удаления «лишних» хромосом как в стационарных, так и в поколенческих генетических алгоритмах основаны на применении оператора редукции, который использует следующие эвристики:

- случайное равновероятное удаление хромосом;
- удаление N_i хромосом, имеющих худшее значение целевой функции;
- удаление хромосом на основе обратно пропорционального значения целевой функции;
- удаление хромосом на основе заданной турнирной стратегии.

Можно предложить еще ряд эвристик удаления. Экспериментальные исследования показали, что удаление всех хромосом с худшим значением целевой функции может привести к преждевременной утрате разнообразия пространства поиска и, следовательно, к попаданию в локальный оптимум. Оставление в популяции большого числа хромосом с «плохим» значением целевой функции приведет к «слепому» поиску.

3. Для количественной оценки шаблонов в генетическом алгоритме используются две характеристики: порядок шаблона — $\mathrm{o}(H)$; определенная длина шаблона — $\delta(H)$.

Порядок шаблона — число закрепленных позиций (в двоичном алфавите это число единиц и нулей, представленных в шаблоне). Например, для шаблона $H = (0 * * * * *)$ порядок шаблона $\mathrm{o}(H) = 1$, а для шаблона $H = (0 * 1 * *)$ величина $\mathrm{o}(H) = 2$.

Определенная длина шаблона — это расстояние между первой и последней позициями нулей и единиц. Например, для шаблона $H = (011 * 1 * *)$, $\delta(H) = 5 - 1 = 4$, а для шаблона $H = (0 * * * * * *)$, $\delta(H) = 1 - 1 = 0$.

При выполнении оператора рекомбинации хромосомы копируются пропорционально значению их целевой функции или, более точно, хромосома i получает выбор с вероятностью $\mathrm{Pr}_i(\mathrm{OP})$, определяемой выражением:

$$\mathrm{Pr}_i(\mathrm{OP}) = \frac{f_i(x)}{\sum\limits_{i=1}^{N_p} f_i(x)}.$$

Пусть имеется некоторый шаблон H, который присутствует в популяции P^t, тогда обозначим $m(H,t)$ число хромосом популяции P^t, которые могут быть получены из шаблона H. После получения набора непересекающихся популяций размера N_p с перемещением части хромосом из популяции P^t в популяцию P^{t+1} предполагается получить $m(H,t+1)$ представителей шаблона H в генерации популяции P^{t+1}. Здесь t означает номер поколения или условный параметр времени.

Тогда согласно известной формуле Д. Холланда выражение $m(H,t+1)$ запишется

$$m(H,t+1) = m(H,t)N_p f(H) \Big/ \sum_{i=1}^{L} f_i(x), \tag{3.4}$$

где $f(H)$ — среднее значение целевой функции хромосом, представленных шаблоном H в генерации t.

Если обозначить среднее значение целевой функции всей популяции как

$$\overline{f}(x) = \sum_{i=1}^{N_p} f_i(x)/N_p,$$

то

$$m(H,t+1) = m(H,t)f(H)/\overline{f}(x).$$

Из полученного выражения видно, что величина $m(H,t+1)$ изменяется как отношение среднего значения целевой функции шаблона к среднему значению целевой функции популяции.

Пусть целевая функция шаблона остается выше среднего значения целевой функции популяции на величину $c \times \overline{f}(x)$, где $c = 1, 2, \ldots$ — коэффициент, постоянный на протяжении одного поколения. Тогда имеем:

$$m(H, t+1) = m(H, t)(\overline{f}(x) + c \times \overline{f}(x))/\overline{f}(x) = (1+c)m(H, t).$$

Начиная работу генетического алгоритма с $t = 0$ и считая $c > 0$ константой, получаем уравнение

$$m(H, t) = m(H, 0)(1+c)^t.$$

Данное выражение показывает, что с течением времени число «хороших» хромосом популяции экспоненциально растет, а число хромосом, имеющих значение целевой функции ниже среднего в популяции, экспоненциально уменьшается.

Существует ***правило репродукции Д. Холланда***, которое формулируется так: *в простом генетическом алгоритму шаблон со значением целевой функции выше среднего копируется в следующую генерацию, а шаблон с целевой функцией ниже среднего — устраняется.*

4. Отметим, что если хромосомы из предыдущей популяции копируются в новую популяцию без обмена генов, то поисковое пространство обычно не увеличивается и процесс эволюционного поиска затухает. Поэтому во всех генетических алгоритмах кроме оператора рекомбинации используются также кроссинговер, мутация и другие генетические операторы. Они создают новые хромосомы и увеличивают или уменьшают количество шаблонов в популяции.

Рассмотрим влияние оператора кроссинговера на возможность выживания шаблонов на следующих генерациях. Для реализации оператора кроссинговера и определения точек разрыва внутри хромосомы используется $(L-1)$ точек. Шаблон в генетическом алгоритме обычно выживает, когда точка оператора кроссинговера попадает вне «определенной длины». Поэтому вероятность выживания для простого оператора кроссинговера записывается так:

$$\Pr_1(s) = 1 - \delta(H)/(L-1), \tag{3.5}$$

где L — длина хромосомы, т. е. суммарная длина входящих в нее генов.

Например, рассмотрим хромосому $P = (01111000)$ и два шаблона $H_1 = (**1***0*)$ и $H_2 = (***11***)$. Шаблон H_1 имеет определенную длину: $\delta(H) = 7 - 3 = 4$. Если точка операторы кроссинговера выбирается равновероятно среди $8 - 1 = 7$ мест, то вероятность разрушения шаблона равна $4/7$ или он выживает с вероятностью $\Pr_1(s) = 3/7$. Аналогично, шаблон H_2 имеет определенную длину $\delta(H) = 5 - 4 = 1$ и разрушается с вероятностью $1/7$, а выживает с вероятностью $\Pr_1(s) = 6/7$.

Для модифицированных операторов кроссинговера, выполняемых с вероятностью $\Pr(\text{ОК})$ вероятность выживания шаблона определится таким образом:

$$\Pr_2(s) \geqslant 1 - (\alpha \Pr(\text{ОК})\delta(H)/(L-1)), \tag{3.6}$$

где α — коэффициент, задающий тип модифицированного оператора кроссинговера, $\alpha = 0{,}1; 0{,}2; 0{,}3; \ldots; 1$. Допуская независимость оператора рекомбинации и оператора кроссинговера, получим объединенную оценку их действия:

$$m(H, t+1) \geqslant [m(H,t)f(H)/\overline{f}(x)][1 - \alpha \Pr(\text{ОК})\delta(H)/(L-1)].$$

Из данного выражения следует, что шаблоны хромосом со значением целевой функции выше среднего и малым значением определенной длины $\delta(H)$ имеют возможность экспоненциального роста в новой популяции.

Рассмотрим влияние модифицированных операторов мутации на возможность выживания хромосом с лучшим значением целевой функции на следующих генерациях. Для того чтобы после применения оператора мутации шаблон H хромосомы выжил в следующей генерации, все фиксированные позиции этого шаблона должны сохраниться. При мутации одна позиция сохраняется с вероятностью $1 - \Pr(\text{ОМ})$. Шаблон H выживает, когда каждая из $\text{о}(H)$ закрепленных позиций сохраняется. Вероятность выживания шаблона при реализации оператора мутации определится $(1 - \Pr(\text{ОМ}))^{\text{о}(H)}$. При малых значениях $\Pr(\text{ОМ}) \ll 1$ вероятность выживания шаблона может быть аппроксимирована выражением

$$\Pr_3(s) = 1 - \text{о}(H)\beta \Pr(\text{ОМ}), \tag{3.7}$$

где β — коэффициент, задающий вид применяемого модифицированного оператора мутации, связанный со значением $\text{о}(H)$ следующим образом: значение β имеет порядок $1/\text{о}(H)$. Например, если $\text{о}(H) = 10^2$, то $\beta = k10^{-2}$. Значение k варьируется таким образом, чтобы значение выражения (3.8) сохранялось в интервале $[0, 1]$. Вероятность применения оператора мутации в простых генетических алгоритмах обычно меняется в пределах от 0,001 до 0,01. Отметим, что в реальных практических задачах порядок $\text{о}(H)$ меньше 10^3. Тогда предполагается, что ожидаемое число копий шаблона H в следующей генерации после реализации модифицированных операторов рекомбинации, кроссинговера и мутации определяется следующим образом:

$$m(H, t+1) > [m(H,t)f(H)/\overline{f}(x)] \times \\ \times [1 - \alpha \Pr(\text{ОК})\,\delta(H)/(L-1) - \text{о}(H)\beta \Pr(\text{ОМ})]. \tag{3.8}$$

По аналогии с фундаментальной теоремой Холланда это выражение назовем модифицированной фундаментальной теоремой генетических алгоритмов.

5. Рассмотрим влияние других операторов на значение $m(H, t+1)$. Вероятность выживания шаблона H в следующих генерациях после применения оператора инверсии запишется так:

$$\Pr_4(s) \geqslant 1 - \varphi \Pr(\text{ОИ}) \cdot \left[\frac{\delta(H)}{L-1} - \frac{\delta(H)^2}{(L-1)^2} \right], \tag{3.9}$$

где $\Pr(\text{ОИ})$ — вероятность выбора хромосомы, соответствующей шаблону H из популяции на заданном шаге генерации (в практических задачах вероятность оператора инверсии обычно находится в интервале $[0; 0{,}5]$); φ — коэффициент, определяющий применение модифицированных операторов инверсии, $\varphi \approx (0{,}1 \div 0{,}9)$. Тогда выражение (3.8) с учетом оператора инверсии запишется так:

$$m(H, t+1) > [m(H,t) f(H)/\overline{f}(x)] \times \\ \times |1 - \alpha \Pr(\text{ОК}) \delta(H)/(L-1) - \text{o}(H) \beta \Pr(\text{ОМ})) - \Pr_4(s)|.$$

Данное выражение перепишем в упрощенном виде:

$$m(H, t+1) > [m(H,t) f(H)/\overline{f}(x)] \, |\Pr_2(s) - \Pr_3(s) - \Pr_4(s)|.$$

При использовании в генетическом алгоритме оператора сегрегации вероятность выживания шаблона на следующей генерации определяется по формуле

$$\Pr_5(s) \geqslant 1 - [\gamma \Pr(\text{ОС}) \delta(H)/(L-1)]/N_p, \tag{3.10}$$

где γ — коэффициент, определяющий применение модифицированных операторов сегрегации, $\gamma \approx 1{-}3$. В практических задачах вероятность оператора сегрегации обычно находится в интервале $[0; 1]$.

Для оператора транслокации вероятность выживания запишется так:

$$\Pr_6(s) \geqslant [1 - \alpha \Pr(\text{ОК}) \, \delta(H)/(L-1)] \, \varphi \Pr(\text{ОИ}).$$

Для оператора транспозиции вероятность выживания запишется следующим образом:

$$\Pr_7(s) \geqslant [1 - \alpha \Pr(\text{ОК}) \, \delta(H)/(L-1)] \, \eta \Pr(\text{ОС}),$$

где η — коэффициент, определяющий применение модифицированных операторов транспозиции, $\eta \approx (0{,}5 \div 1)$.

При использовании оператора удаления вероятность выживания шаблона на следующей генерации определяется по формуле

$$\Pr_8(s) \geqslant [1 - \delta \Pr(\text{ОУ}) \, \delta(H)/(L-1)]/N_p,$$

где δ — коэффициент, определяющий применение оператора удаления, $\delta \approx (0{,}1 \div 0{,}5)$.

При использовании оператора вставки вероятность выживания шаблона на следующей генерации определяется так:

$$\Pr_9(s) \geqslant [1 - \lambda \Pr(\text{ОВ})\, \delta(H)/(L-1)]/N_p,$$

где λ — коэффициент, определяющий применение оператора вставки, $\lambda \approx (0{,}1 \div 0{,}5)$.

Тогда модифицированная фундаментальная теорема генетического алгоритма с учетом всех рассмотренных генетических операторов для решения инженерных оптимизационных задач примет вид

$$m(H, t+1) > [m(H,t) f(H)/\overline{f}(x)]\, |\Pr_2(s) - \Pr_3(s) - \Pr_4(s) - \\ - \Pr_5(s) - \Pr_6(s) - \Pr_7(s) - \Pr_8(s) - \Pr_9(s)|. \quad (3.11)$$

6. Основная теорема генетического алгоритма, приведенная Д. Холландом, показывает асимптотическое число шаблонов, «выживающих» при реализации простого генетического алгоритма на каждой итерации. Выражение (3.11), в отличие от теоремы Холланда, показывает асимптотическое число хромосом, «выживающих» при реализации модифицированных операторов кроссинговера, мутации, инверсии, сегрегации, транслокации, транспозиции, удаления и вставки на каждой генерации. Очевидно, что это число — приблизительное. Оно меняется в зависимости от вероятности применения того или иного оператора. Особенное влияние на число «выживающих» и «умирающих» хромосом при реализации генетического алгоритма оказывает значение целевой функции отдельной хромосомы и всей популяции. Во многих инженерных задачах имеются специальные знания, позволяющие строить упрощенные модели. Коэффициенты α, β, φ, γ, δ, λ, η определяются путем экспериментов, причем их значение подбирается таким образом, что значения $\Pr_2(s) - \Pr_9(s)$ не являются отрицательными. Отметим, что ряд авторов считает, что значения $\Pr_3(s) - \Pr_9(s)$ надо не вычитать, а перемножать. Следует ометить, что эта оценка приближенная и вычитание или перемножение величин $\Pr(s)$ в принципе не оказывает решающего значения на величину реального количества схем, переходящих в следующую генерацию.

Рассмотрим, например, хромосому длины $L = 7$ и два шаблона, представляющие ее:

$$\begin{array}{lccc|cccc} P_i = & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ H_1 = & * & 1 & * & * & * & * & 0 \\ H_2 = & * & * & * & 1 & 0 & * & * \end{array}$$

где | — символ, обозначающий точку кроссинговера. Шаблон H_1 после одноточечного оператора кроссинговера скорей всего будет уничтожен

потому, что 1 в позиции 2 и 0 в позиции 7 попадут в разные хромосомы-потомки. Ясно, что шаблон H_2 сохраняется, так как после реализации оператора кроссинговера число 1 в позиции 4 и 0 в позиции 5 попадут в одного и того же потомка. Хотя точка оператора кроссинговера выбрана случайно, ясно, что шаблон H_1 менее приспособлен к выживанию, чем H_2. Так как точка оператора кроссинговера выбирается случайно среди $L-1=7-1=6$ возможных позиций, то шаблон H_1 уничтожается с вероятностью $\Pr(d)=\delta(H_1)/(L-1)=5/6$. Этот же шаблон (H_1) выживает с вероятностью $\Pr_1(s)=1-\Pr(d)=1/6$. Аналогичным образом, шаблон H_2 имеет $\delta(H_2)=1$ и вероятность его уничтожения $\Pr(d)=1/6$, а вероятность выживания $\Pr_1(s)=5/6$.

7. Вернемся к примеру Д. Голдберга вычисления максимума функции $f(x)=x^2$. Дополнительно пусть имеются три частных шаблона $H_1=(1****)$, $H_2=(*10**)$ и $H_3=(1***0)$, описанные в табл. 3.4.

Таблица 3.4

Шаблон	Перед репродукцией	Номера хромосом	Значение средней ЦФ шаблона $f(H)$
H_1	$1****$	2; 4	348,5
H_2	$*01**$	3; 4	245
H_3	$1***0$	2	16

Во втором и в третьем столбцах табл. 3.5 приведем ожидаемое и реальное число копий хромосом, переходящих с генерации t на генерацию $t+1$ после оператора репродукции. В пятом и шестом столбце приведены те же данные, но после выполнения всех генетических операторов. В четвертом и седьмом столбцах табл. 3.5 приведены номера хромосом.

Таблица 3.5

Схемы	После репродукции			После всех операторов		
	Ожидаемое число	Реальное число	Номера хромосом	Ожидаемое число	Реальное число	Номера хромосом
H_1	3,20	3	2; 4	3,13	3	2; 4
H_2	2,18	2	3; 4	2	2	1; 2
H_3	1,97	2	2	1,15	1	4

Рассмотрим сначала шаблон H_1. В течение репродукции хромосомы копируются с вероятностью, определенной согласно величине их целевой функции. Из первой строки табл. 3.4 следует, что хромосомы с номерами 2 и 4 являются представителями шаблона H_1 (см. табл. 3.2). После оператора репродукции получены две копии шаблона и хромо-

сомы с номерами 2 и 4 вошли в популяцию. Проверим, соответствует ли это число фундаментальной теореме. Из (3.4) при $\alpha, \beta = 1$ ожидается получить $m(H,t) = f(H)/\overline{f}(x)$ копий. Вычисляя среднее значение целевой функции $f(H_1)$, получим $(256 + 441)/2 = 348{,}5$. Разделив это число на среднее значение целевой функции популяции ($\overline{f}(x) = 222{,}5$) и умножая на число H_1 шаблонов в момент времени t ($m(H,t) = 2$), получим ожидаемое число H_1 шаблонов в момент времени $(t+1)$, т. е. $m(H, t+1) = 2 \cdot 348{,}5/222{,}5 = 3{,}13$. Сравнивая это число с реальным числом копий (шаблонов), равным 3, видно, что получили реальное число копий.

Продолжая первый шаг, определим, что оператор кроссинговера не может дать дальнейший эффект, так как определенная длина $\delta(H_1) = 1$ предотвращает разрыв единственного бита. Далее, если взять оператор мутации (ОМ) таким, что $\Pr(\text{ОМ}) = 0{,}01$, то ожидается $m(H, t+1)\Pr(\text{ОМ}) = 3 \cdot 0{,}01 = 0{,}03$, но нет битового обмена с тремя копиями в трех хромосомах. Как результат, для шаблона H_1 получаем ожидаемое экспоненциальное увеличение числа шаблонов.

Рассмотрим теперь схемы H_2 и H_3 с двумя закрепленными позициями. Шаблон H_2 имеет также две хромосомы в начальной популяции (номера 3, 4) и имеет две копии в следующей генерации. Вычисляем $m(H_2) = 2 \cdot 245/222{,}5 = 2$. Здесь 245 — среднее значение целевой функции шаблона, а 222,5 — среднее значение целевой функции популяции. Для H_3 получим только одну хромосому и $m(H_3) = 1 \times$ $\times\, 16/222{,}5 = 0{,}08$. Здесь число 16 — значение средней целевой функции шаблона.

Заметим, что для шаблона H_2 две копии — это хороший результат. При реализации оператора кроссинговера только в одном случае из четырех возможных $(L - 1 = 5 - 1 = 4)$ шаблон $H_2 = (*0\,|\,1**)$ — разрушается. Как результат, шаблон H_2 выживает с высокой вероятностью. Действительное, для шаблона H_2 имеем $m(H_2, t+1) \approx 2$. Оператор кроссинговера с высокой вероятностью разрушает шаблон H_3, так как $\delta(H_3) = 4$.

8. Д. Холланд считает, что необходимо давать предпочтение шаблонам с лучшим значением целевой функции. Однако точного ответа на вопрос, почему это необходимо делать, не существует, или он каждый раз зависит от конкретной задачи. Другой важный вопрос, сколько новых шаблонов в новой генерации необходимо, сколько реально и сколько полезно для получения эффективного результата?

Количество шаблонов, которое может быть получено из популяции размера N_p при длине хромосом равной L согласно Д. Холланду лежит между 2^L и $N_p \cdot 2^L$. Д. Гольдберг и Д. Холланд отмечают, что число «эффективных» шаблонов составляет ориентировочно величину N_p^3.

Рассмотрим популяцию N_p хромосом длины L, представленных в двоичном алфавите. Будем считать только те шаблоны, которые выживают с вероятностью больше, чем константа $\Pr(s)$. Для упроще-

ния рассмотрим простой оператор кроссинговера и оператор мутации с малой вероятностью выполнения. Обычно в хромосоме длины L выделяется участок L_s, равный $L/2$. Это приведет к рассмотрению только тех шаблонов, где $\varepsilon < 1 - \Pr(S)$ и длина $L_s < \varepsilon(L-1)+1$.

Ясно, что существует 2^{L_s-1} таких шаблонов. Для вычисления общего числа шаблонов используется заданный шаблон, который перемещается вдоль хромосомы на каждом шаге поиска. Выполним этот шаг $L - L_S + 1$ раз. Тогда общее число шаблонов Θ длины L_s определится следующим образом:

$$\Theta \leqslant 2^{Ls-1}(L - L_s + 1).$$

Данная формула показывает число шаблонов для одной хромосомы. Очевидно, для целой популяции число шаблонов ориентировочно определяется как

$$\Theta_p \leqslant N_p 2^{L_s-1}(L - L_s + 1).$$

Если взять размер популяции равным $N_p = 2^{L_s/2}$, то нижняя граница числа шаблонов, «выживающих» после работы генетического алгоритма, ориентировочно равна

$$N(s) \geqslant N_p \cdot (L - L_s + 1) \cdot 2^{L_s-2}.$$

Если учесть, что $N_p = 2^{L_s/2}$, то получим

$$N(s) = \frac{\left[(L - L_s + 1) \cdot N_p^3\right]}{4}.$$

Следовательно, число шаблонов, выживающих после одной генерации генетического алгоритма, пропорционально кубу размера популяции и имеет порядок $N(s) = \omega N_p^3$, где ω — коэффициент ($\omega = 1 - 4$).

3.6. Введение в аксиоматическую теорию генетических алгоритмов

1. Сформулируем описание генетических алгоритмов в виде научной теории. Предварительно приведем ряд основных определений этого фундаментального термина.

Теория (от греческого рассмотрение, исследование) — система основных идей в той или иной отрасли знаний, научное объяснение объективных закономерностей развития мира и различных форм движения материи и обобщение объективных результатов, наблюдений, экспериментов.

Теория — систематизированное и логически последовательное воспроизведение в мышлении объективной реальности. В теории объективная логика действительности воспроизводится в строго последовательной системе понятий.

Теория — система принципов, законов, категорий, понятий, концепций, описывающая какое-либо относительно однородное, целостное явление или совокупность его элементов, функций. Критерии научной теории: относительно завершенная логическая структура (принципы, категории, понятия и др.), наличие теоретических конструктов (концепций, положений, гипотез и др.), наличие положений, доказательств, соединяющих эти конструкты с имеющимися фактами, другими теориями.

Теория — греческое умозрение. В науке — объяснение явлений на основании общего принципа, из которого эти явления выводятся, цепь логических построений, проверенных на опыте.

Принцип — основное исходное положение какой-либо теории учения науки, руководящая идея, основное правило исследования и т. п.

Концепция — система идей, взглядов на предмет, явления, способ их понимания, трактовки, определяющая характер познавательной и практической деятельности.

Концепция — система взглядов на процессы и явления в природе. Ведущий замысел, определяющий стратегию действий при осуществлении программ, проектов, планов.

Концепция — логически умственное построение системы воззрений о каком-либо предмете.

2. В общем виде теория есть произвольное множество предложений некоторого языка. Для обозначения высказываний, образованных применением n-местного отношения φ к предметам используется запись $\varphi(t_1, t_2, \ldots, t_n)$, где φ — n-местный предикат, $t_1, t_2, \ldots, t_n$ — термы.

Задавая язык конкретной математической теории (например, генетический алгоритм) можно непосредственно определить формулы и их смысл. Для задания элементарных формул необходимо определить предикаты, используемые в теории и термы этой теории. Для задания термов определяют сорта объектов, константы и операции. Все это в совокупности согласно Н. Непейводе составляет словарь или сигнатуру теории. Задав сигнатуру теории, необходимо проинтерпретировать все понятия, перечисленные в ней. После интерпретации формулы, не содержащие переменных, оказываются либо истинными, либо ложными. Формулы, содержащие переменные становятся истинными либо ложными после задания значений переменных.

На практике интерес представляют не все интерпретации данной сигнатуры, а те, на которых выполнены заданные аксиомы.

Согласно определению Н. Непейводы, теория Th — множество замкнутых формул сигнатуры. Если $A \in \mathrm{Th}$, то A аксиома Th. Модель Th — интерпретация, в которой все аксиомы истинны. Формула A является теоремой (или следствием) теории Th, если A — истинна во всех моделях Th. B — следствие A, если B — теорема $\{A\}$.

3. Отметим, что способ построения научной теории, в основе которой используются исходные положения, называемые аксиомами, а все

остальные предложения теории получаются как логические следствия аксиом, называется ***аксиоматическим методом***. ***Аксиома*** — положение, принимаемое без доказательств в качестве исходного, отправного для данной теории. Основным в нем является метод интерпретаций. Тогда для генетических алгоритмов можно построить следующую базовую теорию.

Пусть каждому исходному понятию и отношению аксиоматической теории генетических алгоритмов поставлен в соответствие некоторый конкретный математический объект. Совокупность таких объектов называется ***полем интерпретации***. Всякому утверждению U теории генетических алгоритмов ставится в соответствие некоторое высказывание U^* об элементах поля интерпретации, которое может быть истинным или ложным. Тогда можно сказать, что утверждение U теории генетических алгоритмов соответственно истинно или ложно в данной интерпретации. Поле интерпретации и его свойства сами обычно являются объектом рассмотрения другой теории простого генетического алгоритма, которая в частности может быть аксиоматической. Этот метод позволяет доказывать суждения типа: если теория генетических алгоритмов непротиворечива, то непротиворечива и теория простого генетического алгоритма.

Пусть теория генетических алгоритмов проинтерпретирована в теории простого генетического алгоритма таким образом, что все аксиомы A_i теории генетических алгоритмов интерпретируются истинными суждениями A_i^* теории простого генетического алгоритма. Тогда всякая теорема теории генетических алгоритмов, т. е. всякое утверждение A, логически выведенное из аксиом A_i в генетическом алгоритме, интерпретируется в простом генетическом алгоритме некоторым утверждением A^*, выводимым в простом генетическом алгоритме из интерпретаций A_i^* аксиом A_i и, следовательно, истинным.

4. Метод интерпретаций позволяет также решать вопрос о независимости систем аксиом: для доказательства того, что аксиома A теории генетических алгоритмов не зависит от остальных аксиом этой теории, т. е. не выводима из них, и, следовательно, необходима для получения всего объема данной теории, достаточно построить такую интерпретацию генетического алгоритма, в которой аксиома A была бы ложна, а все остальные аксиомы этой теории истинны. Уточнением понятия аксиоматической теории является понятие ***формальной системы***. Это позволяет представлять математические теории как точные математические объекты и строить общую теорию или метатеорию таких теорий. Всякая формальная система строится как точно очерченный класс выражений — формул, в котором некоторым точным образом выделяется подкласс формул, называемых теоремами данной формальной системы. При этом формулы формальной системы непосредственно не несут в себе содержательного смысла. Их можно строить из произвольных знаков и символов. Общая схема построе-

ния произвольной формальной системы генетического алгоритма такова:

1. Язык системы генетического алгоритма: аппарат алгебры логики; теория множеств; теория графов, теория алгоритмов, основные положения биологии и теории систем.

   (a) Алфавит — перечень элементарных символов системы: двоичный, десятичный, буквенный, Фибоначчи и др.

   (b) Правила образования (синтаксис), по которым из элементарных символов строятся формулы теории генетического алгоритма:

   - построение моделей эволюций;
   - конструирование популяций;
   - построение целевой функции;
   - разработка генетических операторов;
   - репродукция популяций;
   - рекомбинация популяций;
   - редукция;
   - адаптация.

   Последовательность элементарных символов считается формулой тогда и только тогда, когда она может быть построена с помощью правил образования.

2. Аксиомы системы генетических алгоритмов. Выделяется некоторое множество конечных формул, которые называются аксиомами системы. В генетическом алгоритме существует большое число наборов аксиом. Например, базовый набор аксиом следующий:

   - Популяция конструируется случайным образом.
   - Выполнение оператора репродукции производится на основе «колеса рулетки».
   - Обязательное использование операторов кроссинговера и мутации.
   - Размер популяции после каждой генерации остается постоянным.
   - Размер популяции задается экспертной системой, внешней средой или лицом, принимающим решения.
   - Число копий (решений), переходящих в следующую генерацию, определяется согласно теореме генетических алгоритмов.
   - Целевая функция определяется на основе принципа «выживание сильнейших».

3. Правила вывода генетических алгоритмов. Фиксируется конечная совокупность предикатов $\Pi_1, \Pi_2, \ldots, \Pi_k$ на множестве всех формул системы. Пусть $\Pi(x_1, \ldots, x_{n_i+1})$ — какой-либо из этих предика-

тов ($n_i > 0$), если для данных формул $F_1, \ldots, F_{n_i+1}$ утверждение $\Pi(F_1, \ldots, F_{n_i+1})$ истинно, то говорят, что формула F_{n_i+1} непосредственно следует из формул $F_1, \ldots, F_{n_i+1}$ по правилу Π_i.

Заданием 1, 2, 3 исчерпывается задание формальной системы генетических алгоритмов как точного математического объекта. При этом степень точности определяется уровнем точности задания алфавита, правил образования и правил вывода. Выводом системы генетических алгоритмов называется всякая конечная последовательность формул, в которой каждая формула либо является аксиомой системы генетических алгоритмов, либо непосредственно следует из каких-либо предшествующих ей (этой последовательности) формул по одному из правил вывода Π_i системы.

5. Всякую конкретную математическую теорию генетических алгоритмов можно перевести на язык подходящей формальной системы таким образом, что каждое ложное или истинное предложение теории генетических алгоритмов выражается некоторой формулой системы. Метод интерпретаций позволяет устанавливать факт относительной непротиворечивости, т. е. доказывать суждения типа: «если теория генетических алгоритмов непротиворечива, то непротиворечива и теория простого генетического алгоритма». В общем случае проблема непротиворечивости не решена и является одной из основных в математике.

Итак, можно выстроить следующую цепочку построения и реализации теории генетических алгоритмов:

стратегия–теория–концепция–принципы–аксиомы–гипотезы–практическая реализация

Гипотеза (от греч. предположение) — основание, предположение, выдвинутое с целью объяснения причин, свойств и существования явлений действительности.

Гипотеза — форма развития научных знаний, представляющая собой обоснованное предположение, выдвигаемое с целью объяснения причин свойств и существования явлений действительности. Под гипотезой понимают опирающиеся на конкретные данные предположения.

Предлагается ряд основных стратегий взаимодействия методов эволюционного и локального поиска:

- «***поиск–эволюция***»;
- «***эволюция–поиск***»;
- «***поиск–эволюция–поиск***»;
- «***эволюция–поиск–эволюция***».

Заметим, что иерархически можно строить стратегии такого типа любого уровня сложности. Например, «эволюция–поиск–эволюция–поиск–эволюция–поиск» и т. д. Отметим, что такое построение зависит от наличия вычислительных ресурсов и времени, заданного на получение окончательного решения.

В первом случае любым из описанных алгоритмов поиска или их комбинаций определяется одно или пара альтернативных решений задачи. На основе этих решений строится популяция, к которой применяется одна из схем эволюции. Далее процесс продолжается итерационно до достижения критерия остановки.

Во втором случае конструируется популяция и реализуется одна из схем эволюции. Лучшее решение анализируется и улучшается (если это возможно) одним из алгоритмов поиска. Далее процесс выполняется, как в первом случае. В остальных случаях процесс поиска результатов выполняется аналогично.

6. Приведем основные принципы, которые эффективно используются при генетическом поиске.

Принцип целостности. В генетических алгоритмах значение целевой функции альтернативного решения не сводится к сумме целевых функций частичных решений.

Принцип дополнительности. При решении оптимизационных задач генетическими алгоритмами возникает необходимость использования различных не совместимых и взаимодополняющих моделей эволюций и генетических операторов.

Принцип неточности. При росте сложности анализируемой задачи уменьшается возможность построения точной модели. Здесь используется теория нечетких графов.

Принцип соответствия. Язык описания исходной задачи должен соответствовать наличию имеющейся о ней информации.

Принцип разнообразия путей развития. Реализация генетических алгоритмов многовариантна и альтернативна. Существует много путей эволюции. Основная задача — найти точку бифуркации и выбрать путь, приводящий к получению оптимального решения.

Принцип единства и противоположности порядка и хаоса. «Хаос не только разрушителен, но и конструктивен», т. е. в хаосе области допустимых решений обязательно содержится порядок, определяющий искомое решение.

Принцип совместимости и разделительности. Процесс эволюции носит поступательный, пульсирующий или комбинированный характер. Поэтому модель синтетической эволюции должна сочетать все эти принципы.

Принцип иерархичности. Генетические алгоритмы могут надстраиваться сверху вниз и снизу вверх.

Принцип «Бритвы Оккама». Нежелательно увеличивать сложность архитектуры генетических алгоритмов без необходимости.

Принцип спонтанного возникновения Пригожина. Генетические алгоритмы позволяют спонтанно генерировать наборы альтернативных решений, среди которых с большой вероятностью может возникнуть оптимальное.

Принцип гомеостаза. Генетические алгоритмы конструируются таким образом, что любое полученное альтернативное решение не должно выходить из области допустимых. Операторы в генетических алгоритмах должны позволять получать реальные решения.

3.7. Выводы

Генетические алгоритмы — поисковые алгоритмы, основанные на механизмах селекции и генетики. Они являются мощной стратегией выхода из локальных оптимумов. Эта стратегия заключается в параллельной обработке множества альтернативных решений, концентрируя поиск на наиболее перспективных из них. Причем периодически в каждой итерации можно проводить стохастические изменения в менее перспективных решениях.

Существуют четыре основных отличия генетических алгоритмов от оптимизационных методов:

- прямое преобразование кодов;
- поиск из популяции, а не из единственной точки;
- поиск через элементы (слепой поиск);
- поиск, использующий стохастические и модифицированные операторы, а не детерминированные правила.

Использование генетических алгоритмов при решении инженерных задач позволяет уменьшить объем и время вычислений и упростить моделирование функций, сократить число ошибок моделирования.

3.8. Контрольные вопросы

1. Поясните смысл понятия «генетические алгоритмы».
2. В чем заключается эволюционный поиск?
3. Приведите основные цели и задачи генетических алгоритмов.
4. Выделите основные отличительные особенности генетических алгоритмов.
5. Приведите основные понятия и определения генетических алгоритмов.
6. Что такое целевая функция в генетических алгоритмах?
7. Перечислите предварительные этапы работы генетических алгоритмов.
8. Каким образом в генетических алгоритмах осуществляется выбор способа представления решения?
9. Как производится разработка операторов случайных изменений в генетических алгоритмах?
10. Какие способы «выживания» решений в генетических алгоритмах вы знаете?
11. Поясните, как создается начальная популяция альтернативных решений?

12. Приведите различные модели размножения, используемые в генетических алгоритмах.
13. Дайте определение понятия принципа и приведите примеры принципов построения генетических алгоритмов.
14. Каким образом определяется эффективность генетического алгоритма?
15. Приведите четыре основных принципа формирования начальной популяции.
16. Дайте определение оператора в алгоритме и генетического оператора.
17. Поясните оператор репродукции.
18. Приведите основные виды операторов репродукции (селекции).
19. Приведите основные стратегии реализации оператора репродукции.
20. Определите понятие «предварительная сходимость алгоритма».
21. Дайте определение оператора кроссинговера.
22. Поясните реализацию простого оператора кроссинговера.
23. Поясните реализацию двухточечного оператора кроссинговера.
24. В чем заключается реализация упорядоченного оператора кроссинговера?
25. Поясните реализацию частично-соответствующего оператора кроссинговера.
26. В чем заключается реализация циклического оператора кроссинговера?
27. Приведите пример работы универсального оператора кроссинговера.
28. Поясните основную идею «жадного» алгоритма.
29. В чем заключается реализация «жадного» оператора кроссинговера?
30. Приведите пример работы «жадного» оператора кроссинговера.
31. Опишите основные операторы мутации.
32. Поясните реализацию простого оператора мутации.
33. В чем заключается реализация оператора инверсии?
34. Поясните реализацию оператора транслокации.
35. В чем заключается реализация оператора транспозиции?
36. Приведите пример работы оператора сегрегации.
37. Поясните реализацию оператора удаления.
38. В чем заключается реализация оператора вставки?
39. Поясните принципы работы оператора редукции.
40. В чем заключается оператор рекомбинации?
41. Приведите примеры операций объединения, пересечения и разности хромосом и популяций.
42. Приведите пример кортежа и декартова произведения популяций.
43. Определите понятия «отношение» и «отношение популяций».
44. Приведите основные свойства отношений.
45. Поясните понятие «нечеткое (расплывчатое) множество».
46. Поясните основные логические операции над популяциями.
47. Охарактеризуйте простой генетический алгоритм.

48. Поясните смысл понятия «шаблон» в простом генетическом алгоритме.
49. Приведите фундаментальную теорему для простого генетического алгоритма.
50. Приведите часть фундаментальной теоремы простого генетического алгоритма для оператора репродукции.
51. Приведите часть фундаментальной теоремы простого генетического алгоритма для операторы кроссинговера.
52. Приведите часть фундаментальной теоремы простого генетического алгоритма для оператором мутации.
53. Приведите пример вычисления выживающих шаблонов на основе фундаментальной теоремы простого генетического алгоритма.
54. Приведите пример построения произвольной формальной системы генетических алгоритмов.
55. Поясните основные стратегии взаимодействия методов эволюционного и локального поиска.

3.9. Упражнения

1. Постройте пример создания начальной популяции на основе метода «одеяла» для нахождения минимума функции $f(x) = x^2$ на интервале $[1-10]$.

2. Постройте пример создания начальной популяции на основе метода «фокусировки» для нахождения минимума функции $f(x) = x^2$ на интервале $[1-50]$.

3. Постройте пример создания начальной популяции на основе комбинированного метода для нахождения минимума функции $f(x) = x^2$ на интервале $[1-10]$.

4. Постройте пример создания начальной популяции на основе метода «дробовика» для нахождения минимума функции $f(x) = x^2$ на интервале $[1-10]$.

5. Для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) выполнить оператор репродукции на основе известных методов селекции при нахождении максимума функции $f(x) = x^2$ на интервале $[1-20]$.

6. Приведите пример использования простого генетического алгоритма (Голдберга) для вычисления минимума функции $f(x) = x^3$ на интервале $[1, 2, 3, 4, 5]$.

7. Покажите пример использования простого генетического алгоритма для вычисления максимума функции $f(x) = x^4$ на интервале $[0, 1, 2, 3, 4]$.

8. Покажите на примере минимума функции $f(x) = x^3$ на интервале $[1, 10]$.

9. Приведите алгоритмы работы для различных операторов репродукции.

10. Приведите алгоритмы работы для различных операторов кроссинговера.

11. Приведите алгоритмы работы для различных операторов мутации.

12. Приведите алгоритмы работы для различных операторов инверсии.

13. Приведите алгоритмы работы для различных операторов транслокации.

14. Приведите алгоритмы работы для различных операторов транспозиции.

15. Приведите алгоритмы работы для различных операторов сегрегации.

16. Приведите алгоритмы работы для различных операторов удаления.

17. Приведите алгоритмы работы для различных операторов вставки.

18. Приведите примеры работы различных операторов кроссинговера для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении максимума функции $f(x) = x^2$ на интервале $[1{-}20]$.

19. Приведите примеры работы различных операторов мутации для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении максимума функции $f(x) = x^2$ на интервале $[1{-}20]$.

20. Приведите примеры работы различных операторов инверсии для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении максимума функции $f(x) = x^2$ на интервале $[1{-}20]$.

21. Приведите примеры работы оператора транслокации для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении минимума функции $f(x) = x^2$ на интервале $[1{-}20]$.

22. Приведите примеры работы оператора транспозиции для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении минимума функции $f(x) = x^2$ на интервале $[1{-}20]$.

23. Приведите примеры работы оператора сегрегации для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении минимума функции $f(x) = x^2$ на интервале $[1{-}20]$.

24. Приведите примеры работы оператора удаления для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении минимума функции $f(x) = x^2$ на интервале $[1{-}20]$.

25. Приведите примеры работы оператора вставки для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении минимума функции $f(x) = x^2$ на интервале $[1{-}20]$.

26. Приведите примеры работы оператора редукции для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении максимума функции $f(x) = x^2$ на интервале [1–20].

27. Приведите примеры работы оператора рекомбинации для популяции $P = \{P_1 - P_4\}$, $P_1 = 1001$ (ЦФ $= 9$), $P_2 = 1100$ (ЦФ $= 12$), $P_3 = 0011$ (ЦФ $= 3$), $P_4 = 0101$ (ЦФ $= 5$) при нахождении максимума функции $f(x) = x^2$ на интервале [1–20].

28. Приведите примеры реализации операций объединения, пересечения и разности произвольных хромосом.

29. Приведите пример декартова произведения популяций $P_I = \{P_1, P_2\}$, $P_{II} = \{P_3, P_4\}$.

30. Приведите пример инверсии хромосомы $H = 1100$.

31. Приведите пример композиции хромосом $H_1 = 1100$, $H_1 = 0101$.

32. Приведите пример отношений рефлексивности, симметричности, транзитивности и эквивалентности между множествами хромосом.

33. Приведите пример соответствия и взаимнооднозначного соответствия между множествами хромосом.

34. Приведите пример нечеткого множества хромосом и основные операции над ними.

35. Приведите схему «репродуктивного плана» (упрощенного алгоритма) Д. Холланда.

36. Приведите схему стандартного генетического алгоритма Л. Девиса.

37. Приведите схему простого генетического алгоритма Д. Гольдберга.

38. Приведите пример построения изоморфных и неизоморфных шаблонов и хромосом для шаблона $H = (*1*0)$.

39. Приведите пример вычисления нижней границы числа шаблонов, «выживающих» после работы генетического алгоритма, причем $N_p = 50$, $L = 9$, $L_s = 3$.

40. Покажите на примере вычисление $m(H, t+1)$ для оператора репродукции.

41. Покажите на примере вычисление $m(H, t+1)$ для оператора кроссинговера.

42. Покажите на примере вычисление $m(H, t+1)$ для операторов репродукции и кроссинговера.

43. Покажите на примере вычисление $m(H, t+1)$ для оператора мутации.

44. Покажите на примере вычисление $m(H, t+1)$ для операторов репродукции и мутации.

45. Покажите на примере вычисление $m(H, t+1)$ для операторов кроссинговера и мутации.

46. Покажите на примере вычисление по формуле $m(H, t+1)$ для всего простого генетического алгоритма.

47. Покажите на примере вычисление вероятности «умирания» $\Pr(d)$.

48. Покажите на примере вычисление вероятности «выживания» $\Pr(s)$.

49. Приведите пример построения формальной системы на основе стандартного генетического алгоритма Девиса.

50. Постройте алгоритм реализации стратегии «эволюция–поиск–эволюция–поиск». В качестве эволюции взять модели Дарвина и Поппера. В качестве поиска — поиск в глубину и метод дихотомии.

51. Приведите пример работы алгоритма реализации стратегии «эволюция–поиск».

Глоссарий к разделу 3

Аксиома — положение, принимаемое без доказательств в качестве исходного, отправного для данной теории.

Аксиоматический метод — способ построения научной теории, в основе которой используются аксиомы, а все остальные предложения теории получаются как логические следствия аксиом.

Аллель — функциональное значение гена.

Взаимнооднозначное соответствие — кортеж длины три, в котором каждому элементу первого множества (популяции) соответствует один и только один элемент другого множества.

Ген — элемент хромосомы (часть закодированного решения).

Генетический алгоритм (***ГА***) — адаптивный поисковый метод, который основан на селекции лучших элементов в популяции.

Генетический оператор — средство отображения одного множества на другое. Другими словами, это конструкция, представляющая один шаг из последовательности действий генетического алгоритма.

Генотип — общая генетическая упаковка.

Гипотеза — форма развития научных знаний, представляющая собой обоснованное предположение, выдвигаемое с целью объяснения причин свойств и существования явлений действительности.

«***Жадный***» ***алгоритм*** делает на каждом шаге локально оптимальный выбор, в надежде, что итоговое решение также окажется оптимальным.

«***Жадный***» ***оператор*** — языковая конструкция, позволяющая создавать новые решения на основе частичного выбора на каждом шаге преобразования локально оптимального значения целевой функции.

Изоморфные шаблоны — варианты, из которых можно получить одинаковые хромосомы или шаблоны.

Инверсия кортежа определяется через инверсию его элементов.

Код Грея — двоичный код, последовательные значения которого отличаются только одним двоичным разрядом.

Концепция — логически умственное построение системы воззрений о каком-либо предмете.

Кортеж — упорядоченное множество.

Локус — позиция гена в хромосоме.

Метод интерпретаций — некоторая теория проинтерпретирована в другую теорию таким образом, что все аксиомы первой теории интерпретируются истинными суждениями второй теории. Тогда всякая теорема первой теории, логически выведенная из аксиом, интерпретируется во вторую теорию некоторым утверждением, выводимым в этой теории из интерпретаций и аксиом первой теории.

Множество — любое объединение в одно целое определенных, вполне различимых объектов из нашего восприятия или мысли.

Неизоморфные хромосомы — различные по виду хромосомы, полученные из одного шаблона.

Нечеткое (***расплывчатое***) ***множество*** — множество, которое имеет нечеткие границы, определяющие принадлежность или не принадлежность элемента множеству.

Объединение двух популяций (***хромосом***) — новая популяция (хромосома), которая состоит из элементов, принадлежащих первой популяции (хромосоме) или второй популяции (хромосоме), или обеим популяциям (хромосомам) одновременно.

Оператор — языковая конструкция, представляющая один шаг из последовательности действий или набора описаний алгоритма.

Оператор вставки — языковая конструкция, позволяющая на основе вставки строительных блоков в хромосомы родителей создавать хромосомы потомков.

Оператор инверсии — языковая конструкция, позволяющая на основе инвертирования родительской хромосомы (или ее части) создавать хромосому потомка.

Оператор кроссинговера — языковая конструкция, позволяющая на основе преобразования (скрещивания) хромосом родителей (или их частей) создавать хромосомы потомков.

Оператор мутации — языковая конструкция, позволяющая на основе преобразования родительской хромосомы (или ее части) создавать хромосому потомка.

Оператор редукции — языковая конструкция, позволяющая на основе анализа популяции после одной или нескольких поколений генетического алгоритма уменьшать ее размер до заданной величины.

Оператор рекомбинации — языковая конструкция, которая определяет, как новая генерация хромосом будет построена из родителей и потомков.

Оператор репродукции (***селекция***) (***ОР***) — процесс, посредством которого хромосомы (альтернативные решения), имеющие более высокое значение целевой функции, получают большую возможность для воспроизводства (репродукции) потомков.

Оператор сегрегации — языковая конструкция, позволяющая на основе выбора строительных блоков из хромосом родителей (или их частей) создавать хромосомы потомков.

Оператор транслокации — языковая конструкция, позволяющая на основе скрещивания и инвертирования из пары родительских хромосом (или их частей) создавать две хромосомы потомков.

Оператор транспозиции — языковая конструкция, позволяющая на основе преобразования и инвертирования выделяемой части родительской хромосомы создавать хромосому потомка.

Оператор удаления — языковая конструкция, позволяющая на основе удаления строительных блоков из хромосом родителей (или их частей) создавать хромосомы потомков.

Определенная длина шаблона — расстояние между первой и последней позициями нулей и единиц.

Отношение — связь между любыми объектами в природе.

Отношение между популяциями — пара, причем упорядоченная, первая компонента которой является новой популяцией, а вторая — областью задания отношения.

Отношение эквивалентности — произвольное отношение, которое одновременно рефлексивно, симметрично и транзитивно.

Пересечение двух популяций — новая популяция, состоящая из элементов, которые принадлежат одновременно первой и второй популяции.

Поколенческие генетические алгоритмы имеют динамический размер популяции в каждой последующей генерации.

Поле интерпретации — совокупность математических объектов, которые поставлены в соответствие каждому исходному понятию и отношению аксиоматической теории генетических алгоритмов.

Популяция — множество альтернативных упорядоченных или неупорядоченных решений (хромосом или особей, или индивидуальностей).

Порядок шаблона — число закрепленных позиций.

Правило репродукции Д. Холланда — в простом генетическом алгоритме шаблон со значением целевой функции выше среднего копируется в следующую генерацию, а шаблон с целевой функции ниже среднего — устраняется.

Предварительная сходимость алгоритма — попадание решения в любой один локальный оптимум при наличии их большого количества.

Принцип — это:

- основное исходное положение какой-либо теории;
- внутренняя убежденность в чем-либо;
- основная особенность работы механизма, устройства и т. п.

Простой генетический алгоритм случайно генерирует популяцию хромосом (альтернативных упорядоченных и неупорядоченных решений). Затем производится копирование хромосом, перестановка их частей и генерация новых хромосом (решений) на основе трех операторов: репродукции, кроссинговера и мутации.

Прямое (***декартово***) ***произведение двух популяций*** — популяция, состоящая из всех кортежей длины 2, первая компонента которых принадлежит первой популяции, а вторая компонента — второй популяции.

Разность двух популяций — новая популяция, состоящая из элементов, которые принадлежат первой популяции и не принадлежат второй.

Соответствие — кортеж длины три, первая компонента которого является подмножеством декартова произведения второй и третьей компоненты (популяций).

Стационарные генетические алгоритмы имеют постоянный размер популяции.

Стратегия — оптимальный набор правил и приемов, которые позволяют реализовать общую цель, достигнуть глобальных и локальных целей решаемой задачи.

Строительные блоки — тесно сплетенные между собой гены (части альтернативных решений), которые нежелательно изменять при выполнении генетических операторов.

Теория — система принципов, законов, категорий, понятий, концепций, описывающая какое-либо относительно однородное, целостное явление или совокупность его элементов. Критерии научной теории: относительно завершенная логическая структура (принципы, категории, понятия и др.), наличие теоретических конструктов (концепций, положений, гипотез и др.). Наличие положений, доказательств, соединяющих эти конструкты с имеющимися фактами, другими теориями.

Фенотип формируется посредством связи генетической упаковки с окружающей средой.

Формальная система представляет математические теории как точные математические объекты и строится как точно очерченный класс выражений — формул, в котором точным образом выделяется подкласс формул, называемых теоремами данной системы.

Хэммингово расстояние — количество позиций с несовпадающими значениями генов в хромосомах, представленных в двоичном коде.

Хромосома — элемент популяции (одно альтернативное упорядоченное или неупорядоченное решение), состоящий из генов.

Целевая функция (функция полезности, приспособленности, пригодности) характеризует определенный уровень качества, каждой хромосомы в популяции. Она используется в генетических алгоритмах для сравнения альтернативных решений между собой.

Шаблон в генетических алгоритмах — подмножество хромосом, содержащих знаки «*» (не имеет значения 0 или 1).

Эволюционный поиск — последовательное преобразование одного конечного нечеткого множества промежуточных решений в другое.

Эволюция популяции — чередование поколений, в которых хромосомы (***родители***) изменяют свои значения так, чтобы каждое новое поколение (***потомки***) наилучшим способом приспосабливалось к внешней среде.

«***Элитизм***» — операция, когда хромосома с наилучшим значением целевой функции сразу копируется в новую популяцию.

Эффективность генетического алгоритма — степень реализации запланированных действий алгоритма и достижение требуемых значений целевой функции.

Список литературы к разделу 3

1. *Алексеев О. В. и др.* Автоматизация проектирования радиоэлектронных средств. — М.: Высшая школа, 2000.
2. *Батищев Д. А.* Генетические алгоритмы решения экстремальных задач. — Воронеж: Изд-во ВГТУ, 1995.
3. *Батищев Д. И.*, *Львович Я. Е.*, *Фролов В. Н.* Оптимизация в САПР. — Воронеж: Изд-во ВГУ, 1997.
4. Большая советская энциклопедия. Т. 29. — М.: Сов. энцикл., 1978.
5. *Ботвинник М. М.* О решении неточных переборных задач. — М.: Сов. радио, 1979.
6. *Букатова И. Л.* Эволюционное моделирование и его приложения. — М.: Наука, 1991.
7. *Вороновский Г. К. и др.* Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. — Харьков: Основа, 1997.
8. *Гладков Л. А.*, *Курейчик В. В.*, *Курейчик В. М.* Генетические алгоритмы / Учебное пособие. Под. ред. В. М. Курейчика. — Ростов-на-Дону: ООО «Ростиздат», 2004.
9. *Горбатов В. А.*, *Горбатов А. В.*, *Горбатова М. В.* Дискретная математика / Учебник для студентов втузов. — М.: АСТ Астрель, 2003.
10. *Емельянов В. В.*, *Курейчик В. В.*, *Курейчик В. М.* Теория и практика эволюционного моделирования. — М.: Физматлит, 2003.
11. *Кормен Т.*, *Лейзерсон И.*, *Ривест Р.* Алгоритмы: построения и анализ. — М.: МЦНМО, 2000.
12. *Корнеев В. В. и др.* Базы данных. Интеллектуальная обработка информации. — М.: Нолидж, 2000.
13. *Курейчик В. В.* Эволюционные, синергетические и гомеостатические методы принятия решений. — Таганрог: Изд-во ТРТУ, 2001.
14. *Курейчик В. М.* Генетические алгоритмы и их применение: — Таганрог: Изд-во ТРТУ, 2002.
15. *Курейчик В. М.* Математическое обеспечение конструкторского и технологического проектирования с применением САПР. — М.: Радио и связь, 1990.
16. *Курицкий Б. Я.* Оптимизация вокруг нас. — Л.: Машиностроение, 1989.
17. *Люггер Д. Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем. — М.: Вильямс, 2003.
18. Математическая энциклопедия. Т. 1, А.–Г. — М.: Сов. энцикл., 1972.
19. *Непейвода Н. Н.* Прикладная логика. — Новосибирск: Изд-во НГУ, 2000.
20. *Норенков И. П.*, *Кузьмик П. К.* Информационная поддержка наукоемких изделий. CALS-технологии. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2002.
21. *Растригин Л. А.* Статистические методы поиска. — М.: Наука, 1968.

22. *Редько В. Г.* Эволюционная кибернетика. — М.: Наука, 2001.
23. *Рутковская Д.*, *Пилиньский М.*, *Рутковская Л.* Нейронные сети, генетические алгоритмы и нечеткие системы. — М.: Горячая линия — Телеком, 2004.
24. Словарь-справочник по педагогике / Под ред. П. И. Пидкасистого. — М.: ТЦ Сфера, 2004.
25. *Хаггарти Р.* Дискретная математика для программистов. — М.: Техносфера, 2003.
26. *Чернилевский Д. И.* Дидактические технологии в высшей школе. — М.: Юнити-Дана, 2002.
27. *Гудман Э. Д.*, *Коваленко А. П.* Эволюционные вычисления и генетические алгоритмы // Обозрение прикладной и промышленной математики. — М.: Изд-во ТВП, 1996.
28. Handbook of Genetic Algorithms. Edited by Lawrence Davis. — USA: Van Nostrand Reinhold, New York, 1991.
29. Holland John H., Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence. — USA: University of Michigan, 1975.
30. Goldberg David E. Genetic Algorithms in Search, Optimization and Machine Learning. — USA: Addison-Westley Publishing Company, Inc., 1989.
31. *Koza J. R.* Genetic Programming. — Cambridge/MA: MIT Press, 1994.
32. Practical Handbook of Genetic Algorithms. — Editor I. Chambers. V. 1. Washington, USA, CRC Press, 1995.
33. Practical Handbook of Genetic Algorithms. — Editor I. Chambers. V. 2. Washington, USA, CRC Press, 1995.
34. Practical Handbook of Genetic Algorithms. — Editor I. Chambers. V. 3. Washington, USA, CRC Press, 1999.

Генетические алгоритмы — поисковые алгоритмы, основанные на механизмах селекции и генетики. Они являются мощной стратегией выхода из локальных оптимумов. Использование генетических алгоритмов при решении инженерных задач позволяет уменьшить объем памяти ЭВМ и время вычислений для нахождения оптимальных решений

4. СОВМЕСТНЫЕ СХЕМЫ ЛОКАЛЬНОГО И ГЕНЕТИЧЕСКОГО ПОИСКА

Догадки в эволюционных исследованиях теперь служат той же цели, что в физике: это количественные рабочие гипотезы, направленные на улучшение экспериментальной работы, проверку, усовершенствование или опровержение теории.

В.-Х. Ли, Д. Грор

4.1. Модифицированные генетические операторы

1. Рассмотрим технологии построения генетических операторов для решения практических задач. Опишем сначала оператор рекомбинации. Имеются три различных механизма его работы:

- использование нескольких альтернативных решений;
- использование комбинированного оператора рекомбинации — скрещивания;
- использование комбинированного оператора рекомбинации — скрещивания в среднем.

Все эти операторы могут работать при любом числе родителей. Хромосомы для реализации оператора рекомбинации выбираются случайно. Такой ***многохромосомный механизм глобальной рекомбинации*** является последствием перебора хромосом. Здесь более двух хромосом-родителей могут принимать участие в построении хромосом-потомков. Отметим, что заметное ускорение поиска наблюдается при переходе от половой модели размножения к неполовой модели, т. е. при использовании глобальной рекомбинации вместо варианта с двумя родителями.

2. Для обобщения и расширения стандартного оператора кроссинговера известен ***сканирующий алгоритм***. Он базируется на процедуре просмотра популяции, когда хромосома-потомок строится слева направо. Модифицированная процедура сканирования имеет следующий вид:

```
BEGIN
   Индивидуализация (маркер на первой позиции каждой
                                                хромосомы)
   For i=1 to i=L
   CHOOSE Выбор j из 1,\ldots , r
   Определение i-й позиции родителя j
   UPDATE (Модификация маркера)
END
```

Эта процедура определяет общие положения рекомбинации нескольких родителей и дает общее определение оператора рекомбинации, частные случаи которого определяются механизмами выбора CHOOSE и модификации UPDATE. В простейшем случае модификация UPDATE может сводиться к перемещению маркера на одну позицию вправо. Сканирование может быть адаптировано к упорядоченному представлению, когда каждый ген хромосомы может быть переставлен. Это гарантирует, что «лучшие» гены будут выбраны из родителей в популяции для получения потомков с оптимальным значением целевой функции. В зависимости от механизма выбора родителей CHOOSE возможны различные версии сканирования: единое сканирование (все родители имеют одинаковую вероятность быть выбранными); сканирование по значению целевой функции, когда выбор выполняется пропорционально значению целевой функции (целевое сканирование) и т. п.

Существует ряд практических задач, когда выбор большого числа родителей (альтернативных решений) обеспечивает лучшие результаты. Заметим, что использование оператора рекомбинации с несколькими родителями позволяет увеличить быстродействие эволюционного поиска. Однако это увеличение не всегда может произойти. Поэтому эффективность поиска изучают на тестовых задачах с контролируемыми параметрами.

3. Качество конкретного генетического алгоритма можно оценить расстоянием $D = (f_1 - f_2)/f_1$, где f_1 — глобальный оптимум, полученный экспериментально (если он известен), а f_2 — лучшее значение, найденное этим алгоритмом. Генетические алгоритмы с операторами высокого уровня имеют большие размеры популяции, что и объясняет их достоинства. Приведем ряд гипотез.

Гипотеза 1. *Использование меньшего числа точек разрыва в генетических операторах приводит к повышению быстродействия генетического поиска.*

Гипотеза 2. *Большие размеры популяции улучшают значение целевой функции в генетических алгоритмах до определенного предела.*

Гипотеза 3. *Использование большего числа родителей приводит к увеличению пространства поиска и повышает вероятность получения оптимального или квазиоптимального результата при увеличении времени поиска.*

Опишем технологии построения модифицированных генетических операторов, используя фрактальные множества, методы одномерного поиска (пассивный, последовательный, дихотомии, Фибоначчи и золотого сечения) и другие методы, описанные выше.

Рассмотрим диагональный оператор кроссинговера, который является обобщением многоточечного оператора кроссинговера. Он создает r потомков от r родителей с выбранными $(r-1)$ точками оператора кроссинговера. Потомок формируется из r частей хромосом родителей, взятых по «диагонали». Например, в популяции $P = \{P_1, P_2, P_3\}$ при реализации диагонального оператора кроссинговера с двумя точками разрыва можно построить следующие четыре хромосомы потомка:

P_1:	1	2	|	3	4	|	5	6
P_2:	a	b	|	c	d	|	e	f
P_3:	x	y	|	z	v	|	s	t
P_4:	1	2		c	d		s	t
P_5:	a	b		z	v		5	6
P_6:	x	y		c	d		5	6
P_7:	3	4		e	f		z	v

Здесь P_1, P_2, P_3 — родительские хромосомы популяции P, а P_4, P_5, P_6, P_7 — хромосомы-потомки, полученные после работы диагонального оператора кроссинговера. Такой оператор является частным случаем оператора сегрегации. Легко убедиться, что для $r = 2$ диагональный кроссинговер совпадает с одноточечным кроссинговером и в некоторых случаях обобщает многоточечный оператор кроссинговера с двумя родителями. Диагональный оператор кроссинговера иногда называют кроссинговером триады.

4. В 1980 г. Б. Мандельброт указал на фрактальную геометрию природы. Согласно Б. Мандельброту эволюционирующие системы имеют фрактальную природу и наличие направленного нестихийного отбора и самосогласованную эволюцию. ***Фрактальные объекты*** самоподобны, т. е. их вид не претерпевает существенных изменений при изменении масштабов их деятельности. Множества, имеющие такую структуру, считаются обладающими геометрической (масштабной) универсальностью. Преобразования, создающие такие структуры, — это процессы с обратной связью с большим числом итераций, когда одна и та же операция выполняется снова и снова, аналогично эволюционным процессам. Здесь результат одной итерации является начальным условием для другой и требуется нелинейная зависимость между результатом и реальным значением.

Такие множества объектов называются ***фрактальными множествами***. Основными примерами таких множеств являются множество Кантора и ковер Серпинского. Эти множества обладают геометрической инвариантностью и называются «множества средних третей».

4.1. Рассмотрим простой вариант построения ***множества Кантора***. Пусть на вещественной оси задан отрезок единичной длины $[0, 1]$. Он делится на три равные части и средняя часть является открытым интервалом (1/3, 2/3 вырезается), как показано на рис. 4.1.

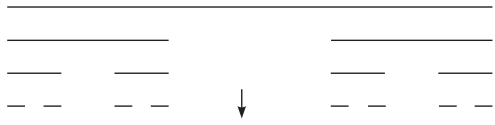


Рис. 4.1. Пример построения множества Кантора

Аналогичные действия выполняются с каждым из оставшихся отрезков. Получаем последовательность отрезков убывающей длины. На первом этапе — это один отрезок, на втором — два, на третьем — 4 и т. д., на k-м — $2k$. При $k \to \infty$ имеем множество точек, называемое множеством Кантора. Суммарная длина всех вырезанных отрезков равна 1.

Множество Кантора, промежуточное между точками $d = 0$ и $d = 1$, является ***фракталом***. Для фракталов введено понятие ***фрактальной размерности***. Для множества Кантора, которое состоит из $N = 2n$ разделенных интервалов длиной $\varepsilon = (1/3)n$, фрактальная размерность равна

$$\text{ФРР} = \frac{n \ln 2}{n \ln 3} \approx 0{,}63.$$

Примером фрактального объекта может быть схема «снежинки». На рис. 4.2 показан процесс построения множества Кантора по шагам. Такой фрактальный объект иногда называют кривой Коха.

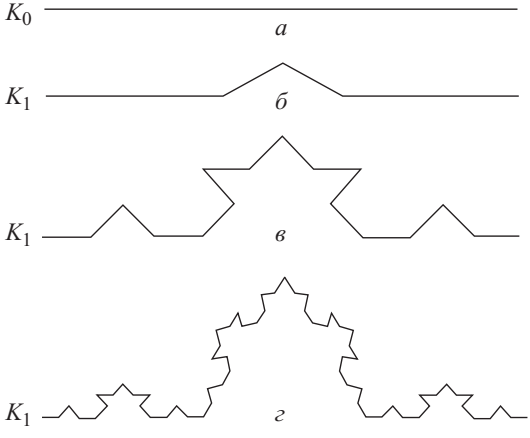


Рис. 4.2. Процесс построения множества Кантора «снежинки»

Каждая треть «снежинки» строится итеративно, начиная с одной из сторон равностороннего треугольника. K_0 — начальный отрезок

(рис. 4.2, *а*). Уберем среднюю треть и добавим два новых отрезка такой же длины. Получим множество K_1 (рис. 4.2, *б*). Повторим процедуру многократно, на каждом шаге заменяя среднюю треть двумя новыми отрезками (рис. 4.2, *в*).

Если взять копию K, уменьшенную в три раза: $r = 1/3$, то все множество K можно составить из $N = 4$ таких копий (рис. 4.2, *г*). Следовательно:

$$\text{ФРР} = \frac{\text{Log}\,(4)}{\text{Log}\,(3)} \approx 1{,}2618.$$

4.2. Обобщение множества Кантора на случай плоских фигур приводит к «***ковру Серпинского***». Например, пусть задано начальное множество S_0 — равносторонний треугольник вместе с областью, которую он занимает (рис. 4.3, *а*).

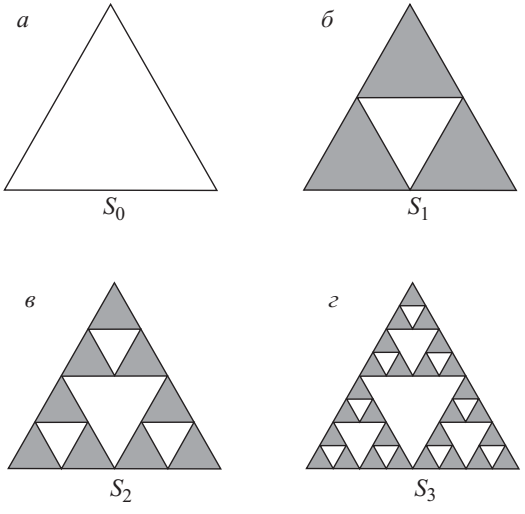


Рис. 4.3. Пример построения ковра Серпинского

Разобьем S_0 на четыре меньшие треугольные области, соединив отрезками середины сторон исходного треугольника. Удалим внутренность центральной треугольной области. Это множество S_1 (рис. 4.3, *б*). Повторив процесс для каждого из трех оставшихся маленьких треугольников, получим S_2 (рис. 4.3, *в*). Продолжая аналогично, получим последовательность вложенных множеств S_n, чье пересечение и образует ковер S (рис. 4.3, *г*).

Весь ковер — объединение $N = 3$ существенно уменьшенных в 2 раза копий: коэффициент подобия $r = 1/2$ (как по горизонтали, так и по вертикали). Следовательно, S — самоподобный фрактал с размерностью

$$\text{ФРР} = \frac{\text{Log}\,(3)}{\text{Log}\,(2)} \approx 1{,}585.$$

Очевидно, что суммарная площадь частей, удаленных при построении, в точности равна площади исходного треугольника. На первом шаге удалили $1/4$ часть площади, на следующем шаге — три треугольника с площадью каждого $(1/4)^2$ и т. д.

Полные доли выкинутой площади:

$$1/4 + 3(1/4)^2 + 3^2(1/4)^3 + \ldots + 3^{n-1}(1/4)^n = 1.$$

Следовательно, оставшееся множество S, т. е. ковер, имеет площадь меры нуль. Отметим, что на основе множества Кантора и ковра Серпинского можно получать любые строительные блоки.

4.3. Опишем оператор кроссинговера на основе идей построения множества Кантора и покажем работу на примере. Пусть заданы две родительские хромосомы P_1, P_2. Согласно идее построения множества Кантора в P_1, P_2 определим две точки разрыва, которые делят хромосому на три равные части $(1/3 + 1/3 + 1/3)$. Если при делении длины хромосомы на 3 ответ не является целым, то берутся ближайшие целые, например, при $L = 7$ получим три части $l_1 = 2$, $l_2 = 2$, $l_3 = 3$ $(l_1 + l_2 + l_3 = 7)$.

Вторая часть хромосомы P_2помещается в хромосому потомок P'_1, а вторая часть хромосомы P_1 помещается в хромосому потомок P'_2:

P_1:	1	2	3	|	4	5	6	|	7	8	9
P_2:	2	4	6	|	8	1	3	|	5	7	9
P'_1:	–	–	–		8	1	3		–	–	–
P'_2:	–	–	–		4	5	6		–	–	–

Дальнейшее заполнение потомка P'_1 выполняется из P_1 слева направо, исключая повторяющиеся и вырезанные гены. Пустые позиции заполняются генами из P_2. Аналогичная процедура выполняется и для построения потомка P'_2. Тогда получим

P'_1:	2	7	9	8	1	3	4	6	5
P'_2:	2	8	7	4	5	6	9	1	3

Продолжая процесс итерационно, в хромосомах P'_1, P'_2 вырежем по два отрезка, расположенных между $[1/9, 2/9]$ и $[7/9, 8/9]$, и произведем перестановку соответствующих генов. При этом получим

P'_1:	2	|	7	|	9	8	1	3	4	|	6	|	5
P'_2:	2	|	8	|	7	4	5	6	9	|	1	|	3
P''_1:	–		8		–	8	1	3	–		1		–
P''_2:	–		7		–	4	5	6	–		6		–

Дальнейшее заполнение потомка P''_1выполняется из P'_1 слева направо, исключая повторяющиеся и вырезанные гены. Пустые позиции

заполняются генами из P_2'. Аналогичная процедура выполняется и для построения потомка P_2''; получаем

P_1'' :	2	8	9	8	1	3	4	$\not{1}$	6
P_2'' :	2	7	9	4	5	6	3	$\not{6}$	1
P_1'' :	2	5	9	8	1	3	4	7	6
P_2'' :	2	7	9	4	5	6	3	8	1

В результате использования оператора кроссинговера множества Кантора получили четыре хромосомы потомка P_1', P_2', P_1'', P_2''.

4.4. Опишем оператор мутации на основе идей построения множества Кантора. Он заключается в перестановке генов, находящихся за точками разреза. Например, пусть задана родительская хромосома P_1. Определим две точки разрыва в P_1, которые делят хромосому на три равные или близкие к ним части:

P_1 :	1	2	3	|	4	5	6	|	7	8	9
P_1' :	1	2	7	|	4	5	6	|	3	8	9

Модификацией оператора мутации множества Кантора является процедура, когда первая часть родительской хромосомы меняется с третьей. Например.

P_1 :	1	2	3	|	4	5	6	|	7	8	9
P_1' :	7	8	9	|	4	5	6	|	1	2	3

Отметим, что на основе множества Кантора и ковра Серпинского можно строить любой генетический оператор. При использовании последовательного поиска выполняется перебор точек разрыва для нахождения хромосомы с оптимальным значением целевой функции.

4.5. Опишем построение генетических операторов с использованием метода дихотомии. Они реализуются за счет механизма перебора точек разрыва.

Приведем нечеткий алгоритм построения оператора кроссинговера дихотомии.

1. Задать две родительские хромосомы длины L.
2. Разделить хромосому (отрезок L) пополам (при нечетном размере хромосомы в любую часть берется большее ближайшее число генов).
3. Определить точку оператора кроссинговера дихотомии с помощью точки разрыва.
4. Получить две новых хромосомы-потомка по правилам построения одноточечного оператора кроссинговера.
5. Разделить каждую половину хромосомы-потомка снова на две части и процесс продолжать аналогично до тех пор, пока не будет

получено заданное количество хромосом-потомков или метод дихотомии завершится. При получении нереальных решений (хромосом с повторяющимися генами) восстановить реальные решения. При этом повторяющиеся гены заменить на отсутствующие гены из хромосом-родителей.

6. Конец работы алгоритма.

Рассмотрим пример. Пусть для реализации оператора кроссинговера дихотомии выбрана популяция P, состоящая из двух родительских хромосом P_1, P_2. Необходимо получить четыре хромосомы-потомков. После первой итерации оператора кроссинговера дихотомии имеем две хромосомы-потомки. Первая хромосома-потомок P_3 состоит из двух строительных блоков: $1СБ_1 \cup 2СБ_2$, здесь $1СБ_1 = (1,2,3,4)$ — первый строительный блок (левая часть родительской хромосомы P_1), а $2СБ_2 = (8,6,7,5)$ — второй строительный блок (правая часть родительской хромосомы P_2). Вторая хромосома-потомок строится аналогично и состоит из двух строительных блоков $2СБ_1 \cup 1СБ_2$ ($2СБ_1 = (4,2,1,3)$, $1СБ_2 = (5,6,7,8)$). На этом первая итерация оператора кроссинговера дихотомии закончена. Вторая итерация выполняется согласно пункту 5 алгоритма. Здесь на втором шаге хромосомы P_3 и P_4становятся родителями. Хромосома P_3 состоит из следующих четырех строительных блоков: $1СБ_3 = (1,2)$, $2СБ_3 = (3,4)$, $3СБ_3 = (8,6)$, $4СБ_3 = (7,5)$, а хромосома P_4 — из блоков $1СБ_4 = (4,2)$, $2СБ_4 = (1,3)$, $3СБ_4 = (5,6)$, $4СБ_4 = (7,8)$. Тогда после применения оператора кроссинговера дихотомии потомок P_5 состоит из строительных блоков $1СБ_3 \cup 3СБ_4 \cup 2СБ_3 \cup 4СБ_4$, а хромосома-потомок P_6 — из блоков $1СБ_4 \cup 3СБ_3 \cup 2СБ_4 \cup 4СБ_3$.

P_1:	1	2	3	4	|	5	6	7	8	P_3:	1	2	|	3	4	|	8	6	|	7	5
P_2:	4	2	1	3	|	8	6	7	5	P_4:	4	2	|	1	3	|	5	6	|	7	8
P_3:	1	2	3	4	|	8	6	7	5	P_5:	1	2		5	6		3	4		7	8
P_4:	4	2	1	3	|	5	6	7	8	P_6:	4	2		8	6		1	3		7	5

Получены четыре хромосомы-потомки P_3, P_4, P_5, P_6. Эффективность метода дихотомии экспоненциально растет с ростом числа хромосом. Количество разбиений в операторе кроссинговера дихотомии назовем глубиной дихотомии. Она обычно задается ЛПР. Отметим, что существует большое количество способов конкретной реализации оператора кроссинговера дихотомии. Его целесообразность определяется на основе вероятности выживания лучших хромосом (альтернативных решений).

4.6. Приведем нечеткий алгоритм реализации оператора мутации на основе дихотомии.

1. Задать родительскую хромосому длины L.
2. Разделить хромосому (отрезок L) пополам (при нечетном размере — в любую часть берется большее число).
3. Определить точку мутации метода дихотомии через точку разрыва.

4. По правилам построения одноточечного оператора мутации получить новую хромосому-потомок.
5. Каждую половину хромосомы-потомка снова разделить на две части и процесс расчета продолжить по исходной схеме до тех пор, пока не будет получено заданное количество хромосом-потомков.
6. Конец работы алгоритма.

Рассмотрим пример работы алгоритма. Пусть задана родительская хромосома P_1. Необходимо выполнить три итерации алгоритма и получить три хромосомы-потомка: P_2, P_3, P_4. Итак, согласно алгоритму, получим

P_1:	1	2	3	4	|	5	6	7	8
P_2:	1	2	3	5	|	4	6	7	8
P_2:	1	2	|	3	5	4	6	|	7	8
P_3:	1	3	|	2	5	4	7	|	6	8
P_3:	1	|	3	2	5	4	7	6	|	8
P_4:	3	|	1	2	5	4	7	8	|	6

Целесообразность оператора мутации дихотомии определяется на основе вероятности его выживания.

4.7. Построение оператора мутации Фибоначчи реализуется за счет аналогичного методу дихотомии механизма перебора точек разрыва. Приведем нечеткий алгоритм построения оператора мутации с использованием метода Фибоначчи. В заданной популяции хромосом на основе селекции выбирается родительская хромосома длины L с наименьшим значением целевой функции.

1. В выбранной хромосоме определить точку разрыва. Она соответствует третьему числу ряда Фибоначчи.
2. Произвести реализацию стандартного оператора мутации и получить новую хромосому-потомок.
3. Вычислить значение целевой функции хромосомы-потомка. Если получен глобальный оптимум, то конец работы алгоритма.
4. Далее в качестве точки оператора мутации Фибоначчи выбрать $4, 5, \ldots$ числа ряда Фибоначчи и перейти к пункту 3. Алгоритм оканчивает работу по достижению заданного критерия или когда номер числа ряда Фибоначчи $\geqslant L$.
5. Конец работы алгоритма.

Рассмотрим пример. Пусть задана родительская хромосома P_1. Используя часть ряда чисел Фибоначчи $(0, 1, 1, 2, 3, 5, 8, \ldots)$, выполним 5 итераций алгоритма. Получим пять хромосом-потомков P_2, P_3, P_4, P_5, P_6. Итак, согласно алгоритму, получим

P_1:	1	|	2	3	4	5	6	7	8	9
P_2:	2	|	1	3	4	5	6	7	8	9

P_2 :	2	1	|	3	4	5	6	7	8	9
P_3 :	2	3	|	1	4	5	6	7	8	9
P_3 :	2	3	1	|	4	5	6	7	8	9
P_4 :	2	3	4	|	1	5	6	7	8	9
P_4 :	2	3	4	1	5	|	6	7	8	9
P_5 :	2	3	4	1	6	|	5	7	8	9
P_5 :	2	3	4	1	6	5	7	8	|	9
P_6 :	2	3	4	1	6	5	7	9	|	8

Целесообразность оператора мутации Фибоначчи определяется на основе знания о решаемой задаче и вероятности выживания лучших решений после его применения.

Отметим, что оператор кроссинговера Фибоначчи практически совпадает с многоточечным оператором кроссинговера. Отличие заключается в том, что точки разреза в операторе кроссинговера Фибоначчи являются фиксированными и соответствуют числам Фибоначчи.

4.8. Рассмотрим построение модифицированных операторов инверсии. Приведем алгоритм построения оператора инверсии с использованием метода Фибоначчи.

1. Задать родительскую хромосому длины L.
2. Точка разрыва оператора инверсии соответствует третьему числу ряда Фибоначчи.
3. По правилам построения оператора инверсии, инвертируя правую часть от точки оператора инверсии, получить новую хромосому потомка.
4. Далее в качестве точки разрыва выбрать числа ряда Фибоначчи $4, 5, \ldots$ и перейти к пункту 3. Алгоритм оканчивает работу по достижению заданного критерия или когда номер числа ряда Фибоначчи $\geqslant L$.
5. Конец работы алгоритма.

Рассмотрим пример. Пусть задана родительская хромосома P_1. Используя часть ряда чисел Фибоначчи $(0, 1, 1, 2, 3, 5, \ldots)$, выполним 5 итераций алгоритма. Получим пять хромосом потомков P_2, P_3, P_4, P_5, P_6. Итак, согласно алгоритму, получим

P_1 :	1	|	2	3	4	5	6	7	8	9
P_2 :	1	|	9	8	7	6	5	4	3	2
P_2 :	1	9	|	8	7	6	5	4	3	2
P_3 :	1	9	|	2	3	4	5	6	7	8

P_3 :	1	9	2	|	3	4	5	6	7	8
P_4 :	1	9	2	|	8	7	6	5	4	3
P_4 :	1	9	2	8	7	|	6	5	4	3
P_5 :	1	9	2	8	7	|	3	4	5	6

Здесь P_1 — родительская хромосома, а P_2, P_3, P_4, P_5 — хромосомы-потомки. Целесообразность оператора инверсии Фибоначчи определяется на основе вероятности его выживания.

4.9. Рассмотрим построение оператора инверсии на основе метода золотого сечения. Это аналогично построению оператора инверсии Фибоначчи. Для оператора инверсии «золотого сечения» предложим следующий механизм реализации. Первая точка разрыва в операторе инверсии определяется на расстоянии целой части $0{,}618L$ от любого края хромосомы. Затем часть хромосомы, расположенная между точкой разрыва и правым концом хромосомы, инвертируется. Вторая точка разрыва в новой хромосоме определяется как ближайшее целое из выражения $(L - 0{,}618L) \cdot 0{,}618$. Далее процесс продолжается аналогично до окончания возможности разбиения хромосомы или по достижению заданного критерия.

Например, пусть задана популяция родительских хромосом (альтернативных решений задачи). В этой популяции выберем хромосому P_1 с наилучшим значением целевой функции и для нее применим оператор инверсии золотого сечения:

P_1 :	A	B	C	D	E	|	F	G	H
P_2 :	A	B	C	D	E	|	H	G	F
P_2 :	A	B	C	|	D	E	H	G	F
P_3 :	A	B	C	|	F	G	H	E	D
P_3 :	A	B	|	C	F	G	H	E	D
P_4 :	A	B	|	D	E	H	G	F	C
P_4 :	A	|	B	D	E	H	G	F	C
P_5 :	A	|	C	F	G	H	E	D	B

Здесь P_1 — родительская хромосома, а P_2, P_3, P_4, P_5 — хромосомы-потомки. Для дальнейшей работы выбираются хромосомы-потомки, у которых значение целевой функции лучше, чем у родительской хромосомы.

4.10. Рассмотрим модифицированные операторы сегрегации. Приведем алгоритм реализации оператора сегрегации золотого сечения.

1. Задать популяцию родительских хромосом длины L.
2. Точки разрыва оператора сегрегации золотого сечения определить как ближайшее целое $0{,}618L$ с обоих концов хромосом.

3. По правилам построения оператора сегрегации в каждой хромосоме случайным или направленным образом выбрать один из трех строительных блоков.
4. Выбранные строительные блоки соединить в хромосому с удалением повторяющихся генов. Далее процесс повторить аналогично.
5. Алгоритм оканчивает работу по достижению заданного критерия или когда проанализированы все строительные блоки.

Рассмотрим пример. Пусть задана популяция P, состоящая из 4 родительских хромосом. Необходимо построить хромосому-потомок на основе оператора сегрегации золотого сечения.

$$
\begin{array}{lccc|cc|ccc}
P_1: & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
P_2: & 2 & 4 & 6 & 8 & 1 & 3 & 5 & 7 \\
P_3: & 1 & 3 & 5 & 7 & 8 & 6 & 4 & 2 \\
P_4: & 8 & 6 & 4 & 1 & 3 & 2 & 5 & 7 \\
\hline
P_5: & 1 & 2 & 3 & 8 & 6 & 4 & 5 & 7
\end{array}
$$

Здесь P_1, P_2, P_3, P_4 — родительские хромосомы популяции P, P_5 — хромосома-потомок. Первая точка разрыва определялись как $0{,}618L = 0{,}618\,8 \approx 5$, а вторая на основе выражения $(L - 0{,}618L) \times$ $\times\, 0{,}618 = (8 - 0{,}618\,8)0{,}618 \approx 3$. Хромосома-потомок P_5 построена из четырех строительных блоков с удалением повторяющихся генов: $\text{СБ}_1(P_1) = (1, 2, 3)$; $\text{СБ}_2(P_2) = (8, 1)$; $\text{СБ}_3(P_3) = (6, 4, 2)$; $\text{СБ}_4(P_4) =$ $= (2, 5, 7)$. Целесообразность оператора сегрегации золотого сечения определяется на основе вероятности его выживания. Отметим, что остальные модифицированные генетические операторы на основе рассмотренных методов строятся аналогично.

5. Важным вопросом при реализации эволюционного поиска является построение целевой функции. Приведем 5 основных методов построения целевой функции.

В первом — значение целевой функции $f(P_i)$ для хромосомы $P_i \in P$ определяется на основе десятичного кодирования. Например, в десятичной системе оценивается стоимость хромосомы.

Во втором значение целевой функции (fP_i) для хромосомы $P_i \in$ $\in P$ определяется на основе двоичного кодирования. Здесь могут быть использованы коды Грея и Хемминга.

В третьем случае используется стандартная целевая функция: $f_c(P_i) = f(\max) - f(P_i)$ для случая максимизации и $f_c(P_i) = f(P_i) -$ $- f(\min)$ для случая минимизации. Здесь $f(\max)$ — наибольшее значение исходной целевой функции; $f(\min)$ — наименьшее значение исходной целевой функции. Чем меньше величина $f_c(P_i)$, тем более пригодна хромосома P_i при решении оптимизационной задачи максимизации.

В четвертом случае строится целевой функции следующего вида:

$$f_{\text{м}}(P_i) = 1/(1 + f_c(P_i)).$$

Значение этой целевой функции лежит в интервале $[0, 1]$.

В пятом методе строится нормализованная целевая функция

$$f_{\text{н}}(P_i) = f_{\text{м}}(P_i) \Big/ \sum_{i=1}^{N_p} f_{\text{м}}(P_i),$$

где N_p — размер исследуемой популяции.

Для построения целевой функции можно использовать любые методы кодирования чисел (троичная, шестнадцатеричная, чисел Фибоначчи и т. п.). Обычно построение целевой функции производится на основе модификаций и различных комбинаций приведенных методов.

6. Отметим, что существуют три основных типа многохромосомных генетических операторов:

- построенные на основе повторения хромосом родителей;
- основанные на сегментировании и рекомбинации хромосом родителей;
- базирующиеся на численном анализе значений целевой функции.

4.2. Архитектуры и стратегии генетического поиска

1. Базовую структуру генетического алгоритма для решения задач оптимизации представим в виде, показанном на рис. 4.4. Здесь блок эволюционной адаптации — специальный блок, который на основе обратных связей управляет процессом эволюционного поиска. Согласно данной схеме на первом этапе случайным, направленным или комбинированным методом получают некоторое подмножество решений

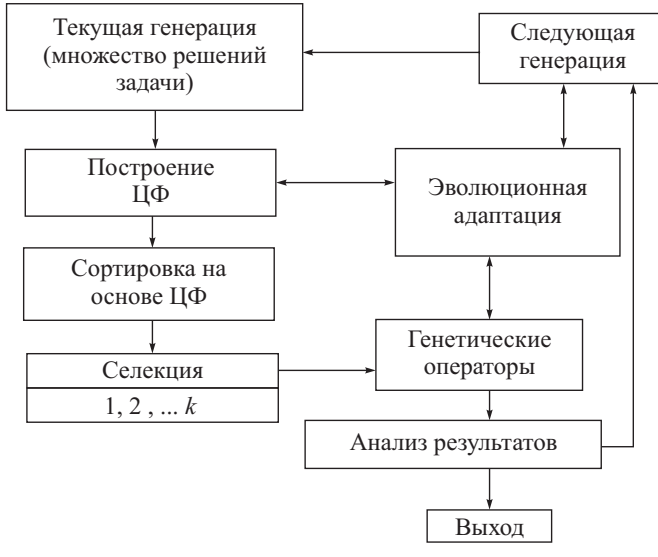


Рис. 4.4. Базовая структура генетического алгоритма

рассматриваемой задачи. Эти решения образуют текущую генерацию или популяцию исследуемых решений на шаге t ($t = 0, 1, \ldots, T$). Далее вводится или вычисляется значение целевой функции. Вычисление целевой функции является сложной задачей, причем от точности значений целевой функции зависит качество будущих решений. ***Качество*** — степень, с которой совокупность присущих решению задачи характеристик удовлетворяет заданным требованиям.

Отметим, что для каждой оптимизационной задачи желательно строить новую целевую функцию. При построении целевой функции необходимо использовать знания о конкретной задаче. На основе целевой функции производятся ранжирование и сортировка популяции решений. Затем в результате различных методов селекции в популяции подбираются родительские хромосомы для применения различных генетических операторов. После реализации всех операторов получается новое подмножество решений P'. Оно объединяется с первоначальным подмножеством решений. Получается новое множество $P_{\text{ГА}} = P \cup P'$. Используя значение целевой функции, производится анализ $P_{\text{ГА}}$. Все элементы в $P_{\text{ГА}}$ (решения задачи), значения целевой функции которых хуже заданного порога, являются с нашей точки зрения неперспективными решениями и удаляются из $P_{\text{ГА}}$. Получается новое множество $P'_{\text{ГА}}$, причем $|P'_{\text{ГА}}| = |P|$. Если данное условие не выполняется, например, $|P'_{\text{ГА}}| < |P|$, то в $P'_{\text{ГА}}$ включается элемент с лучшими характеристиками из отброшенных. Множество $P'_{\text{ГА}}$ объявляется новой текущей популяцией решений и далее процесс может повторяться на основе блока ***эволюционной адаптации*** итерационно до получения подмножества или одного оптимального решения.

Заметим, что такая стратегия позволяет быстрее находить локально-оптимальные результаты. Это связано с параллельной обработкой множества альтернативных решений, причем в такой схеме можно концентрировать поиск на получение более перспективных решений. Отметим, что периодически в каждой итерации проводят различные изменения в перспективных, неперспективных и в других решениях. Временная сложность таких алгоритмов в основном совпадает со сложностью быстрых итерационных алгоритмов и лежит в пределах $O(n) - O(n^3)$, где n — число входов алгоритмов. Такая величина сложности показывает перспективность использования генетических алгоритмов с блоком эволюционной адаптации при решении оптимизационных задач.

2. В настоящее время известно много нестандартных архитектур эволюционного поиска, позволяющих в большинстве случаев частично решать проблему предварительной сходимости алгоритмов. Это методы миграции, метагенетической оптимизации, стохастически-итерационные генетические и поисковые, методы «прерывистого равновесия», объединения генетического поиска и моделирования отжига и др. Отметим, что ***процедура*** — установленный порядок выполнения

какой-либо деятельности или процесса. ***Миграция*** — процедура установления порядка обмена хромосом между популяциями.

Рассмотрим кратко эти методы. Предварительно дадим определение микро-, макро- и метаэволюции. ***Микроэволюция*** — это создание одной хромосомы и реализация на ее основе эволюционного поиска.

Макроэволюция — это создание одной популяции и реализация на ней эволюционного поиска.

Метаэволюция — это создание множества популяций и реализация на нем эволюционного поиска.

2.1. Опишем архитектуру эволюционного поиска с миграцией (рис. 4.5). Здесь предварительно выполняется этап ***метаэволюции***, т. е. создается не одна популяция, а некоторое множество.

Эволюционный поиск осуществляется путем объединения хромосом из различных популяций. Блоки 1–3 в этой схеме представляют собой простой генетический алгоритм. Заметим, что в каждом блоке выполняются различные операторы репродукции. В первом блоке — ОР1 на основе рулетки. Во втором блоке используется ОР2 на основе заданной шкалы. В третьем блоке — ОР3 на основе элитной селекции. В блок миграции каждый раз отправляется лучший представитель из

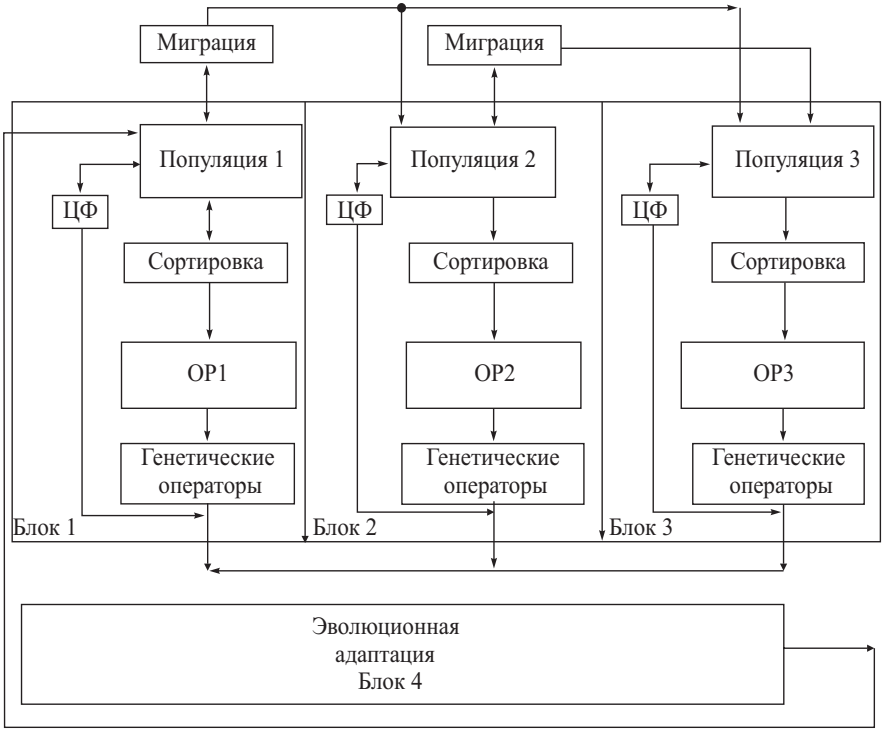


Рис. 4.5. Схема эволюционного поиска на основе миграции

популяции. Связь между блоками 1–3 осуществляется на основе следующих путей: (1–2, 2–3), (1–3). В блоке 4 осуществляется установление баланса на размер популяции на основе эволюционной адаптации и управления всем процессом эволюционного поиска.

Отметим, что можно организовать различное количество связей между блоками по принципу полного графа, по принципу звезды и т. д. ***Граф*** — математический объект, состоящий из множества вершин (точек) и множества ребер (линий), находящихся в заданном отношении. ***Полный граф*** — это граф, в котором любая пара вершин соединена ребром.

2.2. Схема (рис. 4.5) при наличии большого количества вычислительных ресурсов может быть доведена до N блоков. Причем $N - 1$ блоков могут параллельно осуществлять эволюционную адаптацию и через блоки миграции обмениваться лучшими представителями решений. Последний блок собирает лучшие решения, может окончить результат работы или продолжить эволюционный поиск. Для таких схем авторы используют ***платоновы графы***, т. е. правильные многоугольники, которые, как считалось в древних учениях, обладают внутренней красотой и гармонией.

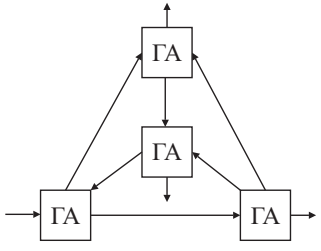


Рис. 4.6, *а*. Схема поиска на основе тетраэдра

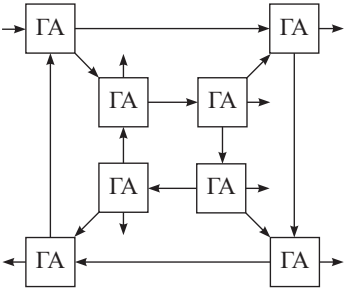


Рис. 4.6, *б*. Схема поиска на основе гексаэдра (куба)

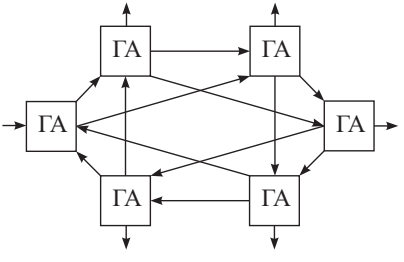


Рис. 4.6, *в*. Схема поиска на основе октаэдра

На рис. 4.6, *а–д* приведены упрощенные схемы организации связей при эволюционном поиске на основе платоновых графов; на рис. 4.6, *а*

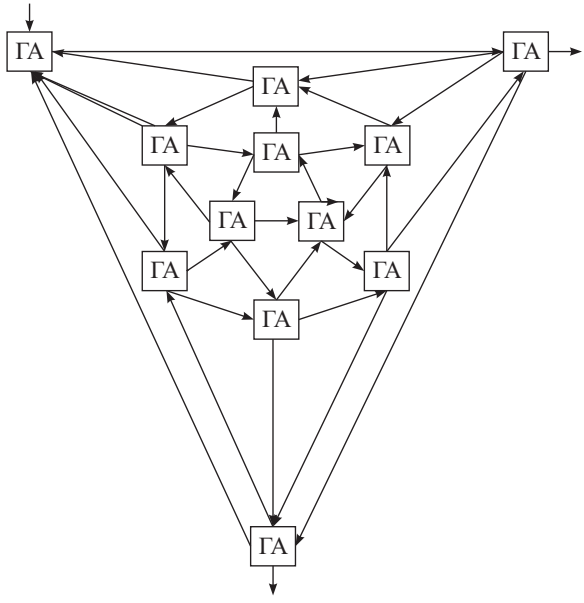


Рис. 4.6, *г*. Схема поиска на основе икосаэдра

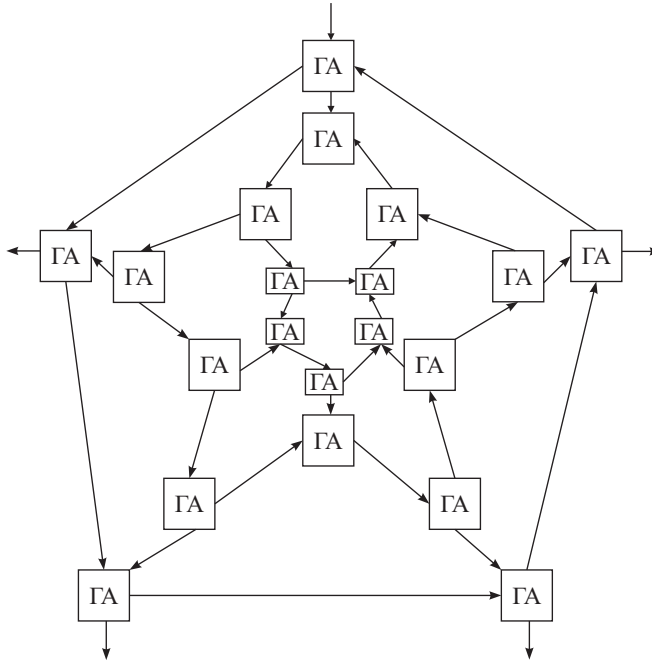


Рис. 4.6, *д*. Схема поиска на основе додекаэдра

на основе тетраэдра; *б* — гексаэдра (куба); *в* — октаэдра; *г* — икосаэдра; *д* — додекаэдра. На рис. 4.6 использовано отношение вход–выход «1–многие». Отметим, что здесь могут быть использованы и другие отношения вход–выход: «1–1»; «многие–1»; «многие–многие». ***Эффективность*** таких отношений проверяется экспериментально. Под эффективностью понимается степень реализации запланированной деятельности и достижения запланированных результатов.

2.3. Существует большое количество и других схем, часть из которых приведена на рис. 4.7, *а–д*. На рис. 4.7, *а* — схема последовательного поиска. Здесь возможно разбиение популяции на подпопуляции с уменьшением времени реализации генетических операторов. На рис. 4.7, *б* — последовательная схема поиска с циклами. Использование такой схемы, с одной стороны, вводит избыточность, а с другой — позволяет расширять область поиска решений.

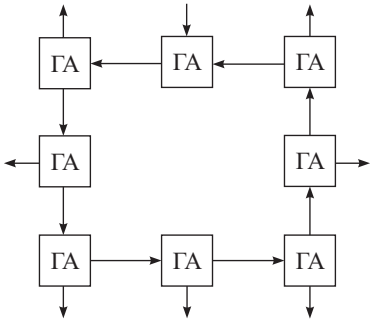


Рис. 4.7, *а*. Схема последовательного поиска

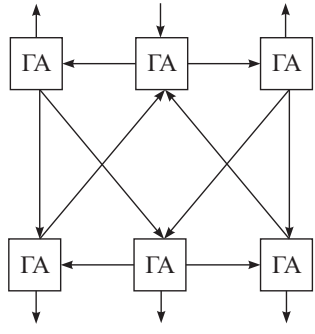


Рис. 4.7, *б*. Последовательная схема поиска с циклами

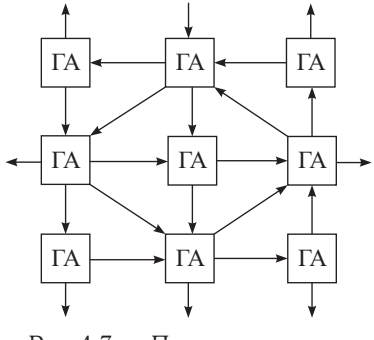


Рис. 4.7, *в*. Последовательная схема поиска на основе треугольников

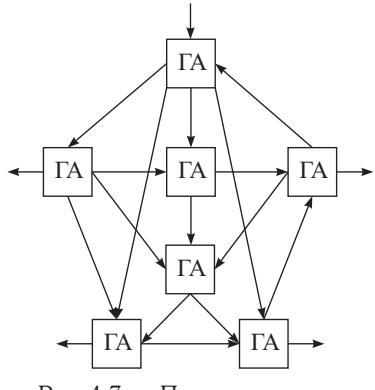


Рис. 4.7, *г*. Поиск на основе модифицированного полного графа на 5 вершин

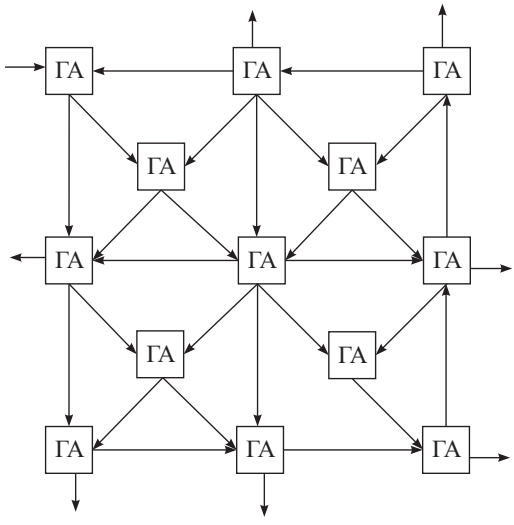


Рис. 4.7, *д*. Последовательная схема поиска на основе ячеечной структуры

На рис. 4.7, *в* — последовательная схема взаимосвязей генетических алгоритмов при эволюционном поиске на основе треугольников, за счет создания устойчивых строительных блоков на высших иерархических уровнях. Введение обратных связей позволяет стабилизировать поиск и осуществлять взаимодействие с внешней средой. На рис. 4.7, *г* — последовательная схема поиска на основе ***полного графа*** на пять вершин. Этот граф также состоит из треугольников. В этой связи такая схема эволюционного поиска позволяет устанавливать баланс и применяется для решения оптимизационных задач. Отметим, что можно строить схемы эволюционного поиска на основе полных графов на любое количество вершин. На рис. 4.7, *д* — последовательная схема поиска на основе ячеечной структуры. Здесь также используются строительные блоки, состоящие из треугольников.

3. Описанные схемы эволюционного поиска, состоящие из нескольких генетических алгоритмов, выполняемых параллельно, последовательно-параллельно и параллельно-последовательно, позволяют во многих случаях выходить из локальных оптимумов.

3.1. На рис. 4.8 показана схема реализации процесса метагенетической оптимизации. Здесь основным является первый блок, в котором осуществляется реализация генетического алгоритма, генерация новых решений, определение значения целевой функции и использование предыдущих решений для генерации лучших результатов. Второй блок позволяет использовать «историю» предыдущих решений для генерации лучшего множества параметров. В третьем блоке генерируется новое множество оптимизационных параметров.

Рис. 4.8. Метагенетический оптимизационный процесс

Используя метагенетический оптимизационный процесс, можно случайным, направленным или случайно-направленным способом генерировать начальные популяции, моделировать их и выполнять генетический алгоритм. Можно случайно выбирать родителей из популяции с произвольной или заданной вероятностью, причем вероятность выполнения каждого оператора может определяться пропорционально значению его целевой функции. Окончательное множество параметров выбирается после моделирования из конечной популяции на основе блока эволюционной адаптации. Отметим, что для каждой оптимизационной задачи желательно строить конкретный метагенетический алгоритм. Такое построение зависит от наличия вычислительных ресурсов и времени, заданного на получение окончательного решения.

3.2. Решение основных оптимизационных задач на основе указанных стратегий осуществляется последовательными, итерационными, точными или комбинированными алгоритмами. В основном эти алгоритмы выполняют совершенствование начального варианта решения, полученного одним из случайных методов. Такой подход, кроме известных преимуществ, обладает трудноразрешимыми недостатками:

- все алгоритмы, кроме точных, часто останавливаются на локальных решениях, которые могут быть далеки от оптимальных, и с трудом осуществляют выход из локальных оптимумов;
- точные алгоритмы могут работать только с очень ограниченным количеством элементов, что для задач большой размерности является недостаточным;
- в качестве исходного используется лишь один вариант решения.
- такие методы чувствительны к выбору начальных условий.

Оптимизационные задачи используют в качестве исходного не одно, а несколько альтернативных решений, причем в зависимости от сложности перерабатываемой информации исходные решения могут быть получены на основе стохастических, детерминированных или комбинированных алгоритмов. Далее полученные решения обрабатываются адаптированными к решаемой задачи генетическими алгоритмами.

На рис. 4.9, *а*, *б* показаны примеры реализации стратегии «поиск–эволюция» и «эволюция–поиск», причем на рис. 4.9, *а* используются два различных генетических алгоритма, а на рис. 4.9, *б* — два различных метода поиска.

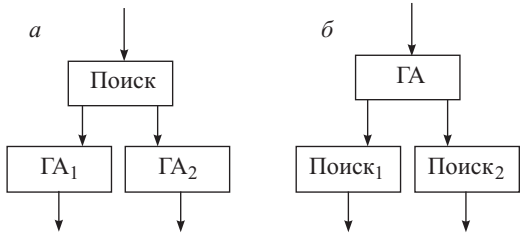


Рис. 4.9. Примеры реализации стратегий

На рис. 4.10, *а*, *б* показаны модификации этой схемы поиска с использованием большого числа генетических алгоритмов.

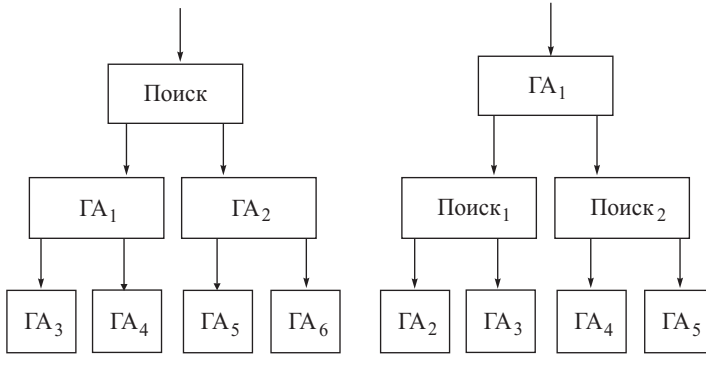


Рис. 4.10. Модификации схем стратегий

На рис. 4.11 приведена горизонтальная схема стратегии «эволюция–поиск–эволюция». Внутри блоков «Поиск$_1$» и «Поиск$_2$» организованы коммутирующие блоки с n входами и n выходами. Это позволяет соединять входы блоков поиска с выходами $n!$ способами, что дает возможность расширять просмотр пространства допустимых решений.

На рис. 4.12 приведена схема реализации стратегий «эволюция–поиск–эволюция–поиск–эволюция». На рис. 4.13 — «эволюция–поиск–эволюция–поиск–эволюция–поиск–эволюция».

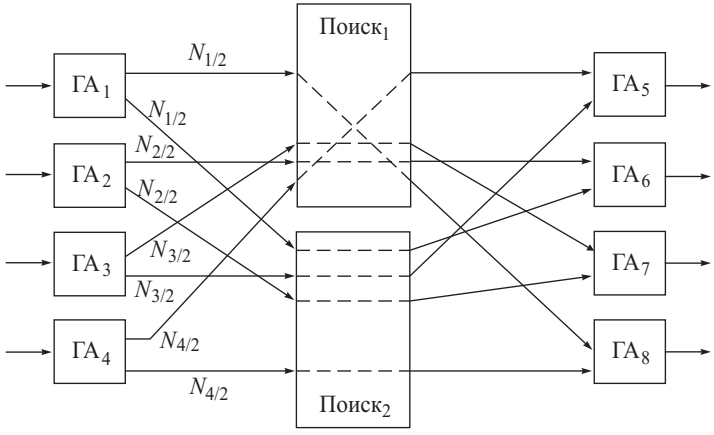


Рис. 4.11. Горизонтальная схема стратегии

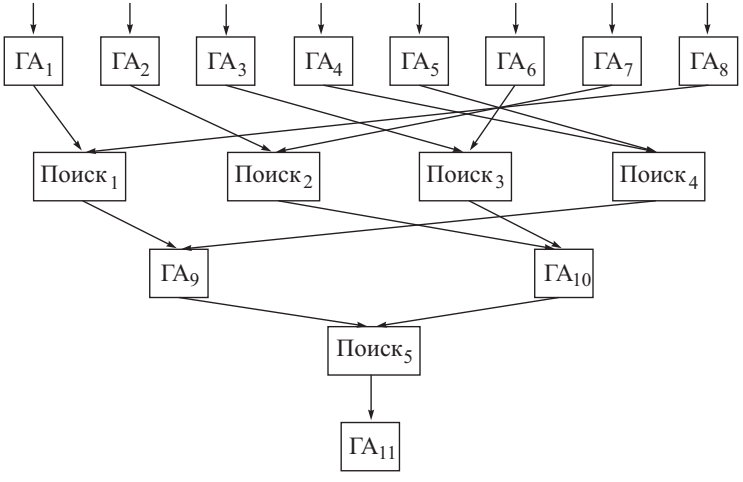


Рис. 4.12. Схема стратегии «эволюция–поиск–эволюция–поиск–эволюция»

На рис. 4.14 показан пример архитектуры соподчинения в эволюционном поиске. Здесь все уровни связаны с блоком внешней среды. Эта структура раскрывает подробно блоки генетических алгоритмов, приведенных на рис. 4.9–4.13.

Генетические алгоритмы могут быть реализованы и другими способами.

3.3. ***Стохастически-итерационный метод*** заключается в следующем. Сначала определяются стартовые точки для направленного поиска. Поиск осуществляется совместно с генетическими операторами на основе блока эволюционной адаптации. После нахождения стартовых точек можно также параллельно использовать методы оптимиза-

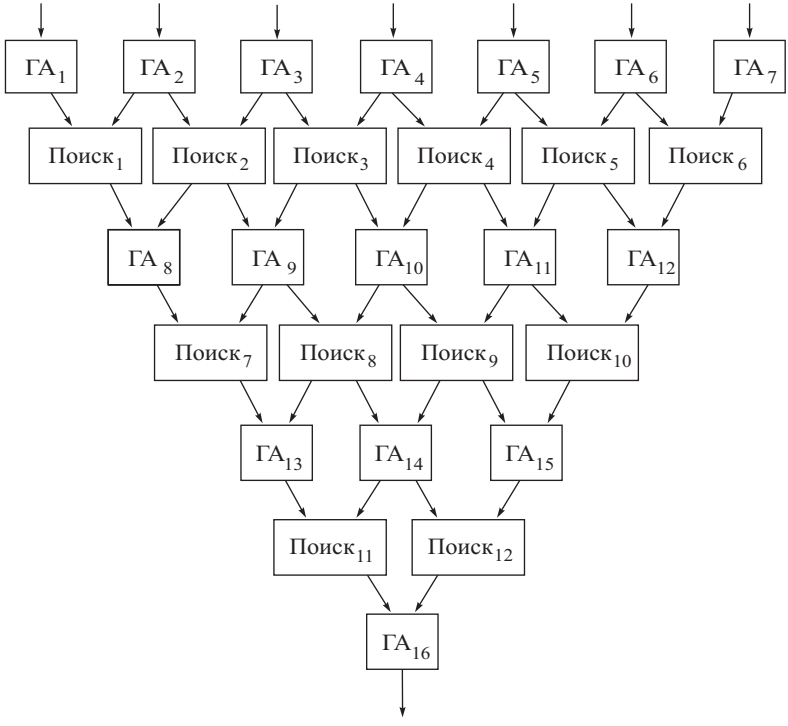


Рис. 4.13. Схема стратегии «эволюция–поиск–эволюция–поиск–эволюция–поиск–эволюция»

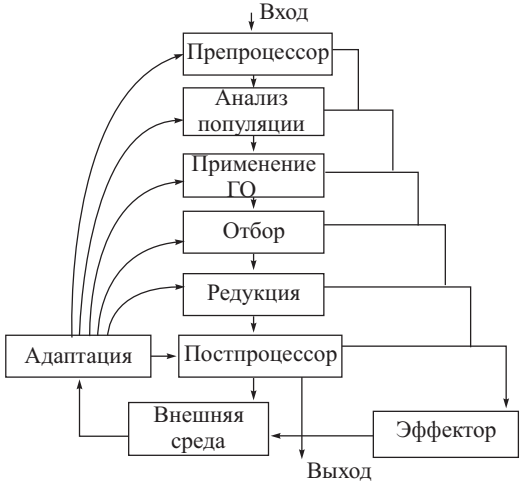


Рис. 4.14. Пример реализации архитектуры соподчинения в эволюционном поиске

ции (метод золотого сечения, градиентного спуска, поиска в глубину и ширину, ветвей и границ и др.).

3.4. ***Метод прерывистого равновесия*** использует палеонтологическую теорию, которая строит модели эволюции на основе описаний вулканических и других изменений земной коры. Он аналогичен моделям эволюции де Фриза. Для применения данного метода предлагается после каждой генерации случайным образом «перемешивать» элементы в популяции, а затем формировать новые текущие популяции с дальнейшей реализацией различных генетических операторов. Здесь можно предложить, как аналог из природы, бессознательный отбор родительских пар и синтетический отбор «лучших» хромосом. Далее случайным образом смешать результаты обоих отборов и не оставлять размер популяции постоянным, а управлять им на основе блока эволюционной адаптации в зависимости от наличия «лучших» элементов. Такая модификация метода прерывистого равновесия позволяет сократить неперспективные популяции и расширить популяции, в которых находятся «лучшие» элементы. Метод прерывистого равновесия — это мощный стрессовый метод изменения окружающей среды, который используется для эффективного выхода из локальных оптимумов.

3.5. Объединение генетического алгоритма и метода моделирования отжига позволяют получать качественные результаты за счет усложнения процедуры оптимизации. Например, на основе простого генетического алгоритма можно получить некоторое подмножество родителей с лучшими характеристиками и для одного из них (наилучшего) или некоторого подмножества применить оптимизационную процедуру моделирования отжига. Такое объединение можно делать различными способами.

3.6. На рис. 4.15 показана упрощенная модель эволюционного поиска при совместном использовании моделей эволюций Дарвина и Ламарка. Первый элемент эволюции Ламарка предусматривает случай, когда ***лицо***, ***принимающее решение*** (***ЛПР***) применяет ее ко всей популяции (шкала = 1). Этот элемент собирает полное множество номеров хромосом в популяции и для них вызывает алгоритм Ламарка. Второй элемент используется, если процентное соотношение в популяции больше 0. Тогда число хромосом для процесса вычисляется по параметру процентного соотношения и размеру популяции и соответственно обрабатывается набор номеров хромосом в популяции. Второй элемент эволюции Ламарка вызывает предикат для выбора множества номеров хромосом, лежащих в пределах от 1 до размера популяции. Для выполнения этого предикат многократно повторяется до тех пор, пока не будет набран требуемый размер. Здесь используется стандартный оператор репродукции для выбора хромосом. Заметим, что если большой процент популяции подвергается эволюции Ламарка, то этот процесс может быть медленным, так как выбор хромосом для данной

эволюции требует дополнительного времени. Если ЛПР не желает использовать эволюцию Ламарка, то шкалу выбираем равную 0.

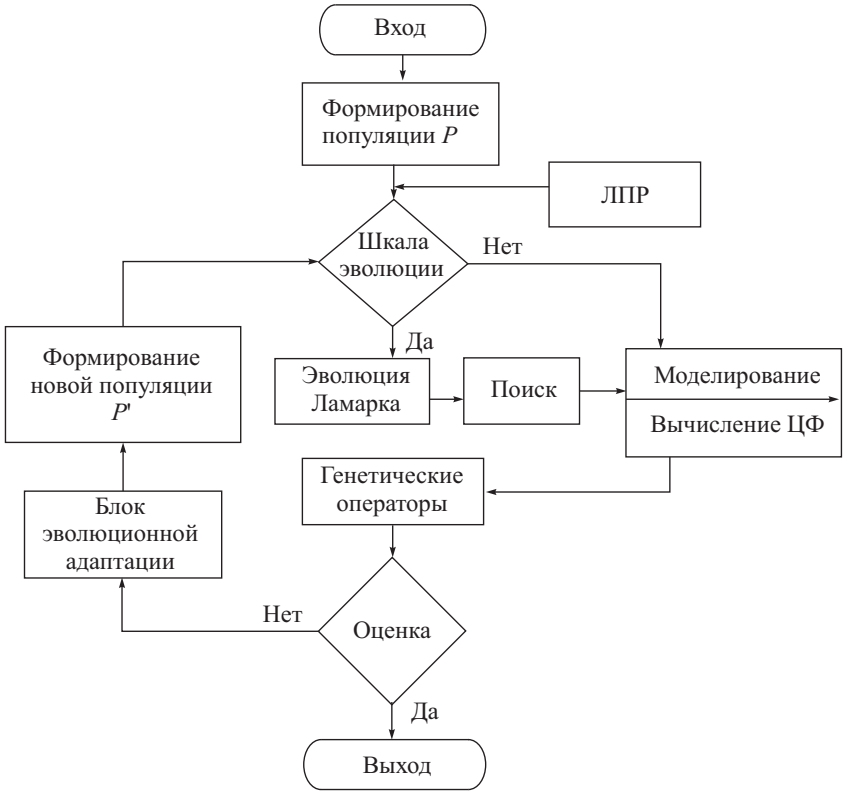


Рис. 4.15. Структурная схема совместного алгоритма эволюций

Кроме основных введен предикат цикла, который контролирует эволюцию Ламарка перечнем выбранных хромосом. Для каждой хромосомы определяются значения целевой функции. Затем выполняется локализованный поиск для заданного числа итераций. Результатом этого поиска является новая хромосома с другим значением целевой функции. Если значение целевой функции лучше, чем у исходной хромосомы, то она заменяется оптимизированным выражением. В противном случае замены не происходит. ЛПР на основе блока эволюционной адаптации может запустить или отменить эволюцию Ламарка. Шаблоном локализованного поиска является метод горизонта, или поиска в глубину. Они анализируют хромосому с лучшим значением целевой функции для поиска лучшего локального оптимума. Отметим, что оператор мутации разрешен для всего поискового пространства. Хромосома с наилучшим значением целевой функции старой и новой популяции используется для последующих генераций.

3.7. В схему совместного алгоритма можно включить любой набор поисковых алгоритмов. Генетический алгоритм использует гибридный подход, в котором генерации определяются для исследования эволюции Ламарка и накопления статистики. Это позволяет частично решать проблему предварительной сходимости алгоритмов. Эффективность использования эволюции Ламарка совместно с эволюцией Дарвина зависит от исследуемой проблемы. Локализованная оптимизация с использованием этих моделей будет повышать среднее значение целевой функции популяции и, следовательно, ускорять выполнение поиска в исходном алгоритме.

3.8. Рассмотрим модифицированные архитектуры поиска. Один из возможных строительных блоков построения многоуровневой архитектуры для решения инженерных задач показан на рис. 4.16. Здесь P — начальная популяция альтернативных решений. На создание новой популяции P' оказывают влияние не только генетический алгоритм и блок эволюционной адаптации, но и внешняя среда. Из таких строительных блоков, как из «кирпичиков», строится архитектура поиска любой сложности.

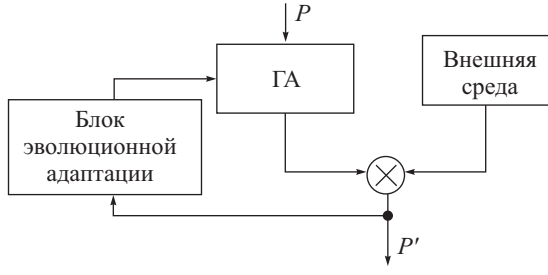


Рис. 4.16. Схема строительного блока

Отметим, что принятие решений в неопределенных, расплывчатых условиях при решении инженерных задач — это генерация возможных альтернативных решений, их оценка и выбор лучшей альтернативы.

4. ***Задачей принятия решений*** называют кортеж $Z = \langle N, \Theta \rangle$ (где N — множество вариантов решений задачи; Θ — принцип оптимальности, дающий представление о качестве вариантов, в простейшем случае — правило предпочтения вариантов). Решением задачи α называют множество $N_{\text{оп}} \subseteq N$, полученное на основе принципа оптимальности. Задачу Z решают следующим образом. Составляют множество N, если это возможно, т. е. определяют варианты, а затем решают задачу выбора. Стандартная система поддержки принятия решения состоит из следующих основных блоков:

- генерация возможных альтернатив решений (сценариев);
- оценки решений (построения целевой функции));
- согласование решений, анализ динамики развития ситуации;

- выбор решения (группы решений), сценария;
- оценка соответствия принятых решений заданным целям.

Сравнивая эту систему со структурой генетических алгоритмов, можно заметить много общего. Следовательно, предложенные концепции, принципы, архитектуры и генетические алгоритмы могут быть использованы в системах поддержки принятия решения.

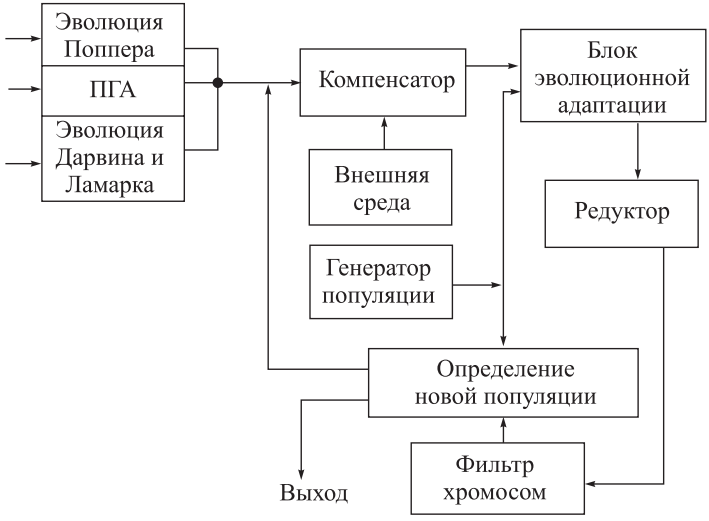


Рис. 4.17. Комбинированная схема принятия решения на основе блока эволюционной адаптации

На рис. 4.17 показана комбинированная схема принятия решения с обратной связью. Здесь ПГА — это простой генетический алгоритм, реализующий модель эволюции Дарвина. Блок эволюции Дарвина и эволюции Ламарка — это модель их совместной реализации. Компенсатор регулирует динамически изменяемый размер популяции решений, расширяя, сужая или оставляя ее постоянной. После реализации эволюций Дарвина и Ламарка компенсатор во взаимодействии с внешней средой реализует принципы эволюционного поиска, при этом лучшие хромосомы выбираются для смешивания популяций и выхода из локальных оптимумов. Сумматор, редуктор и фильтр позволяют повысить эффективность реализации эволюции и скорость принятия решений. Отметим, что в инженерных задачах большой размерности процесс решения усложняется, но выполнение параллельных генетических алгоритмов снижает временную сложность.

Также отметим, что внешняя среда в конечном итоге определяет эволюцию, но не в простой форме связи между наследственной изменчивостью организмов и средой как в эволюции Ламарка, а в более сложной форме.

5. При моделировании возможно объединение всех видов и форм эволюции. На рис. 4.18 показана базовая структура генетического поиска на основе использования моделей эволюций Дарвина, Ламарка, де Фриза. Здесь на основе шкалы эволюции при взаимодействии с внешней средой вырабатываются сигналы 0 — на выполнение эволюции Дарвина; 1 — на реализацию эволюции Ламарка; 0,5 — на реализацию эволюции де Фриза.

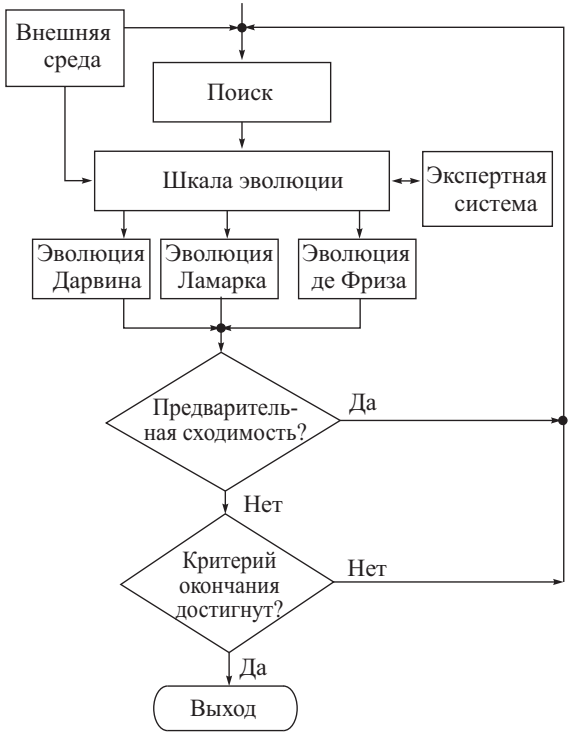


Рис. 4.18. Базовая структура генетического поиска на основе использования эволюций Дарвина, Ламарка и де Фриза

6. Экспертная система позволяет сформулировать набор правил построения целевой функции и окончания поиска. Повторение эволюционного поиска возможно при предварительной сходимости алгоритма или при достижении заданного значения целевой функции. Особенностью данной схемы является использование поисковых стратегий, описанных выше.

На рис. 4.19 приведена модификация базовой структуры генетического поиска на основе использования моделей эволюций Дарвина, Ламарка, де Фриза и Поппера. Здесь, в отличие от базисной структуры, шкала эволюции, взаимодействуя только с внешней средой,

вырабатывает сигналы на выбор эволюции Дарвина (0), Ламарка (1), де Фриза (0,5). После этого выполняется модель эволюции Поппера, реализующая один из видов эвристического поиска в виде метода проб и ошибок.

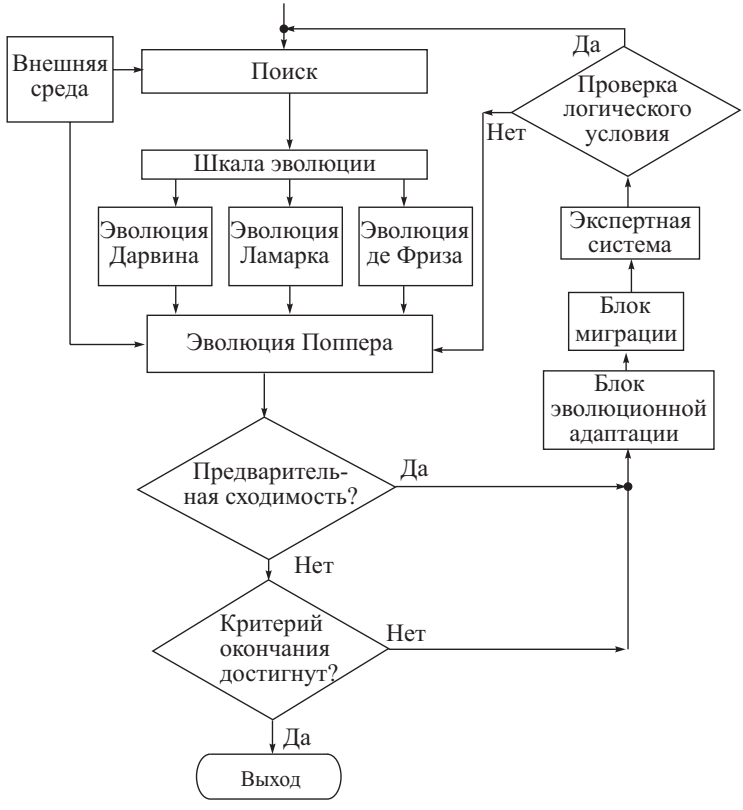


Рис. 4.19. Модификация базовой структуры

6.1. В цепи обратной связи добавлены блоки адаптации и миграции. Они позволяют производить построение порядка из хаоса, установление баланса в системе, выбор параметров для управления эволюционным поиском с целью получения оптимальных и квазиоптимальных решений.

Для повышения качества принятия решений используют, кроме описанных, технологии поиска, связанные с распараллеливанием генетических алгоритмов, структурированием популяции, миграцией хромосом и популяций и т. п. Обычно используются два подхода структурирования популяции.

- После конструирования популяции она случайным или заданным образом разбивается на несколько подпопуляций. Модели эволю-

ции Дарвина, Ламарка, де Фриза, Поппера и Кимуры или любая их модификация применяются внутри каждой подпопуляции. При этом возможно случайное или направленное перемещение хромосом между любыми или заданными популяциями.

- После инициализации популяции для каждой хромосомы определяется ее пространственное местоположение в популяции. Далее, если необходимо, выполняется разбиение популяции. Выбор хромосом для использования генетических операторов зависит от «близости» решений в пространстве (значение целевой функции).

Возможно большое количество комбинаций и модификаций этих подходов с учетом блока эволюционной адаптации, причем эффективность их устанавливается в основном эмпирически.

6.2. Известна двухуровневая система генетического алгоритма Э. Гудмана и Б. Панча (DAGA2). Здесь реализована идея многоуровневой эволюции, когда в генетическом алгоритме используются два уровня. На первом уровне DAGA2 выполняет параллельную реализацию простых генетических алгоритмов, пока они не начнут сходиться. Затем в наилучших решениях (хромосомах) каждой популяции определяются ближайшие друг к другу объекты (гены), которые образуют кластеры. На основе кластеров создается новая хромосома как объект для второго уровня. Отметим, что DAGA2 может действовать как параллельный генетический алгоритм со скрещиванием в локальной области или на основе «островов». В этой архитектуре хромосомы из подпопуляций в грубой (неточной) модели задачи мигрируют в другие подпопуляции в негрубой (точной) модели. Эта структура может быть модифицирована на любое число уровней в зависимости от памяти и временных ограничений на получение результата.

6.3. Известны также параллельные генетические алгоритмы со скрещиванием в локальной области и на основе «островов». В первом случае хромосомы одной популяции моделируются как точки в некотором пространстве. Вводится отношение близости, и генетические операторы применяются к хромосомам, расположенным близко друг от друга. Во втором случае популяции эволюционируют в течение нескольких заданных генераций независимо на определенных «островах», а затем производится обмен генетическим материалом и снова проводится независимая эволюция. Этот процесс происходит в цикле разнообразными способами.

6.4. Я. Парми ввел понятие ***кластерно-ориентированного*** генетического алгоритма, основанного на стратегии быстрого разбиения поисковых пространств на области высоких значений целевой функции. В таком алгоритме установлен режим мутации переменных, введен адаптивный фильтр, отсекающий решения с низким значением целевой функции, причем нижняя граница целевой функции адаптивна в смысле зависимости от решений на каждой генерации. В качестве мо-

дификации такого алгоритма используют начальное структурирование популяции на основе значений целевой функции, реализацию моделей эволюций Дарвина, Ламарка, де Фриза и Поппера в каждом поисковом пространстве. Выбор генетических операторов здесь осуществляется на основе динамической экспертной системы с использованием статистических методов оптимизации.

6.5. Дж. Шаффер и Л. Эшельман предложили ***проблемно-ориентированные*** генетические алгоритмы, вырабатывающие решения (фенотипы) комбинаторных задач, представленных в виде задач параметрической оптимизации, состоящих из строк параметров (генотипов). В данных алгоритмах создается проблемно-ориентированный генератор решений (хромосом). Кроме этого, строится эвристический генератор планов, состоящий из набора шаблонов решений. Далее на основе «жадной» эвристики и значений целевой функции строится возможный план решения задачи. Изменяя параметры, генетический алгоритм заставляет «жадные» эвристики вырабатывать большое число разнообразных хромосом. Взаимодействие генетического алгоритма и эвристического генератора планов позволяет устанавливать время генерации, производить адаптацию наборов параметров, фокусировать наилучшие решения.

6.6. К. Де Йонг предложил в популяциях, имеющих общие хромосомы, выбирать для дальнейшего поиска $N \cdot G$ элементов. Здесь G — число генераций, а N — мощность начальной популяции. Потомки размещаются в новые популяции. Хромосомы из популяции выбираются случайно. Следовательно, при выборе размера популяции нет точных рекомендаций — большая популяция дает потери хромосом; малая популяция дает большую возможность попасть в плохой локальный оптимум.

К. Де Йонг предложил 5 вариантов планирования процесса генетического поиска:

- ***элитная модель***. Пусть $P'(t)$ — лучшая хромосома, сгенерированная за время t. Если после генерации $(t+1)$ $P'(t)$ не попала в $(t+1)$, то $P'(t)$ включается в $(t+1)$ как $N_p + 1$ член;
- ***модель с ожидаемой величиной***. Здесь для селекции выбирается случайная хромосома и вычисляется ожидаемое число потомков для каждой популяции;
- ***элитная модель с ожидаемой величиной***. Это объединение первой и второй моделей;
- «***редукционный фактор***» (CF). Здесь хромосомы-потомки после генерации анализируются на предмет выживания. Хромосомы с целевой функции ниже средней выбираются из подмножества CF и удаляются. Обычно $CF = 2, 3, 4$;
- ***модель генерации новых и модифицированных генетических операторов***.

Одной из простых технологий эволюционного поиска является так называемая «устойчивая репродукция» Л. Девиса (stady-state reproduction). Она соответствует следующему алгоритму:

1. Создать k потомков (хромосом) из начальной популяции через репродукцию.
2. Убрать k членов из начальной популяции, чтобы освободить место для получаемых потомков.
3. Оценить и вставить полученные хромосомы (потомки) на освободившиеся места в популяции.
4. Конец работы алгоритма.

6.7. При оценке хромосом используются два основных подхода: линейная нормализация и определение «окна». В первом случае производится упорядочивание хромосом по убыванию значений целевой функции. Выбирается целевая функция с постоянной величиной и шкала линейного уменьшения ее значений. Во втором случае определяется хромосома с наименьшим значением целевой функции. Хромосомы, у которых значение целевой функции меньше определенного, не участвуют в репродукции. Такой метод показал эффективные результаты на детерминированных проблемах.

6.8. Ю. А. Абилов и др. предложили ***групповой генетический алгоритм с направленной мутацией***. Он состоит из двух уровней. Верхний уровень алгоритма выполняет групповой, а нижний — индивидуальный поиск. На первом этапе берется популяция размером в несколько раз больше, чем в простом генетическом алгоритме, с целью большего охвата всего пространства поиска. Элементы в популяции оцениваются, затем хромосомы со значением целевой функции меньше средней отбрасываются, а из оставшихся хромосом составляются подпопуляции. Далее поиск ведется внутри отдельных групп. Приведем модификацию этого подхода.

1. Сконструировать начальную популяцию P размера $|P| = N_p$, $P_i \in \in P$, $i = \overline{1, N_p}$.
2. Определить значение целевой функции для всех хромосом в популяции и вычислить среднее значение целевой функции по формуле
$$f_{\text{ср}} = \frac{1}{N_p} \sum_{i=1}^{N_p} f(P_i).$$
3. При реализации оператора репродукции (селекции) оставить в популяции хромосомы со значением целевой функции больше средней по всей популяции P_j ($j = \overline{1, N_p}$), $f(P_j) > f_{\text{ср}}$.
4. С помощью шкалы эволюции на основе взаимодействия с внешней средой, блоками адаптации и ЭС выбрать совместную или одну из моделей эволюции Дарвина, Ламарка, де Фриза, Поппера, Кимуры.
5. Образовать группы хромосом по принципу их пространственной близости $|P_i - P_j| \leqslant \delta$, где δ — параметр соседства: $\delta = 2L_k/N_p$, где L_k — длина отрезка, на котором задана k-компонента хромосом

P_i и P_j, при этом хромосомы с «малопригодной» целевой функцией удаляются, и каждая группа как бы исследует отдельный локальный оптимум.

6. Провести новую селекцию, оставив по одной хромосоме в каждой группе разбиения. Проводить генетические операторы над всеми хромосомами, пока не получено заданное решение (если оно известно) или значение критерия остановки алгоритма.
7. Конец работы алгоритма.

6.9. Успех нечеткого генетического алгоритма зависит от правильного выбора размера популяции N_p. Во многих оптимизационных задачах N_p не известно заранее и его можно только прогнозировать, используя знания о характере функции процесса поиска или любые другие знания.

На рис. 4.20 показана укрупненная схема параллельного эволюционного поиска при разбиении популяции на две подпопуляции. Здесь в блоках генетических операторов выполняются операторы кроссинговера, мутации, инверсии, сегрегации, транслокации, удаления и вставки. В блоке редукции производится удаление хромосом со значением целевой функции ниже средней.

Рис. 4.20. Схема параллельного эволюционного поиска

Эволюционный поиск для решения инженерных задач — сложная система и по отношению к ней не может быть использована какая-то одна модель, схема, архитектура, представляющая сложную сеть взаимосвязанных элементов, свойств и функций и находящаяся в определенном взаимодействии с окружающей средой.

4.3. Генетическое программирование

1. Основные исследования в области генетического программирования проведены Д. Коза. ***Генетическое программирование*** представляет собой одно из направлений эволюционного поиска и ориентировано в основном на решение задач автоматического синтеза программ. Хромосомы или структуры, которые автоматически генерируются с помощью генетических операторов, являются компьютерными программами различной величины и сложности.

Программы состоят из функций, переменных и констант. Исходная популяция P хромосом в генетическом программировании образуется стохастически и состоит из программ, которые включают в себя элементы множества проблемно-ориентированных элементарных функций, а также проблемно-ориентированные переменные и константы. Эти множества являются основой для эволюционного синтеза программы, способной наилучшим образом решать поставленную задачу. Одновременно устанавливаются правила выбора элементов из указанных множеств в пространстве всех потенциально синтезируемых программ. Эти множества, а также правила их обработки, оказывают влияние на размерность пространства поиска наилучшего решения и на качество результатов, получаемых методами генетического программирования. Структуры генетического программирования, как правило, имеют древовидную форму.

2. В исследованиях по генетическому программированию применяется язык `LISP`, обладающий необходимыми для синтеза структур свойствами:

- `LISP` является синтаксически простым функциональным языком, программа на котором представляет собой рекурсивную функцию символьных выражений, состоящих из элементарных функций, условных операторов и операторов суперпозиции;
- обработка данных в `LISP`-программе сводится к объединению, делению и перегруппировке информации;
- `LISP`-выражения представляются древовидной структурой (рис. 4.17), форма и величина которой может динамически изменяться.

Следуя Д. Коза, сформулируем следующие стартовые условия для генетического программирования:

- установить множество проблемно-ориентированных переменных и констант (`terminal set`);

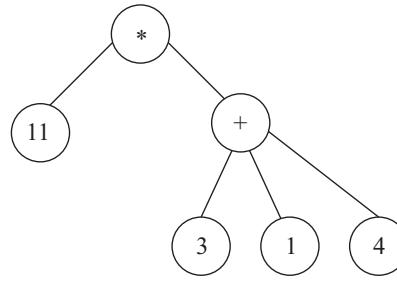


Рис. 4.17. Дерево LISP-выражения (∗11(+3 1 4))

- установить множество проблемно-ориентированных элементарных функций (`function set`);
- определить экстремальные значения целевой функции;
- установить параметры моделей эволюций;
- определить критерий остановки моделирования эволюции и правил декодирования результатов эволюции.

Поскольку основой моделирования эволюции в генетическом программировании являются элементы множеств `terminal set` и `function set`, то, в этом смысле, выбор пользователем языка программирования будет в дальнейшем определять вид получаемых решений. Здесь определение значений целевой функции, параметров эволюции и критериев остановки процесса моделирования совпадает с аналогичными этапами в других типах генетических алгоритмов.

В качестве элементов `function sets` могут фигурировать следующие:

- арифметические операции (например, $+$, $-$, $*$);
- математические функции (например, sin, cos);
- булевы операции (например, `if-then-else`);
- циклы (например, `for, do-until`);
- специальные функции для быстрого поиска хороших решений.

Элементами множества `terminal set` являются константы и переменные, среди которых особое значение имеют так называемые случайные константы с коротким временем «жизни». Множества `function set` и `terminal set` должны быть достаточными для нахождения решения задачи, а любая функция — корректно выполнимой при любых допустимых аргументах.

3. Общепризнанным способом оценки качества целевой функции является такой показатель, как среднеквадратичная ошибка (чем она меньше, тем лучше программа). Иногда используется критерий «выигрыша», согласно которому выигрыш определяется в зависимости от степени близости к корректному значению целевой функции. Целевую функцию в генетическом программировании обозначают через F_{ro}. На практике используется стандартное значение F_{st}. Обозначим через a_i некоторую программу из популяции размером N_p. Тогда стандартное

значение целевой функции определяется так:

$$F_{\mathrm{st}}(a_i) = \begin{cases} F_{\mathrm{ro}}(a_i), & \text{если наименьшее значение } F_{\mathrm{ro}}(a_i) \\ & \text{является подходящим,} \\ F_{\max} - F_{\mathrm{ro}}(a_i), & \text{если наибольшее значение } F_{\mathrm{ro}}(a_i) \\ & \text{является подходящим,} \end{cases}$$

а юстируемое значение целевой функции равно $F_{jN}(a_i) = 1/(1 + F_{\mathrm{st}}(a_i))$.

Интервал изменения $F_{jN}(a_i)$ равен $(0, 1)$. Размер популяции N_p в генетическом программировании обычно составляет несколько тысяч программ. Рекомендации о максимальном числе генераций $t_{\max}$ отсутствуют.

4. Рассмотрим основные этапы генетического программирования.

1. *Инициализация*. На этом этапе стохастически генерируется популяция P, состоящая из N_p древовидных программ, причем корневой вершиной дерева всегда является функция, аргументы которой выбираются случайно из множеств `function set` или `terminal set`. Концевыми вершинами дерева должны быть переменные или константы, в противном случае процесс генерации необходимо рекурсивно продолжить. Если структура дерева становится сложной, то заранее устанавливается максимальная высота дерева, равная числу ребер дерева, которое содержит самый длинный путь от корневой вершины до некоторой концевой вершины.
2. *Оценка решений*. На втором этапе оценивается значение целевой функции каждой программы. Так как программы выбраны случайно, то для большинства из них значения целевой функции будут отличаться от лучшего решения, т. е. для оценки можно взять разницу между лучшим и худшим значением целевой функции в популяции.
3. *Генерация новой популяции*. Этот этап принято разделять на следующие подэтапы.

   3.1. Выбор операторов генетического программирования. Основными операторами здесь являются репродукция и кроссинговер, применяемые с вероятностями $\Pr(\mathrm{OP})$ и $\Pr(\mathrm{OK})$ соответственно, причем $\Pr(\mathrm{OP}) + \Pr(\mathrm{OK}) = 1$ (чаще всего $\Pr(\mathrm{OP}) = 0{,}1$, $\Pr(\mathrm{OK}) = 0{,}9$).

   3.2. Селекция и рекомбинация. Данный этап моделирования выполняется по схемам, аналогичным генетическому алгоритму.

   3.3. Образование новой популяции. Если к некоторой программе применяют оператор репродукции, то эта программа копируется в новую популяцию. Для проведения оператора кроссинговера выбираются две родительские хромосомы (программы), случайным образом определяются точки кроссинговера и путем обмена частей родительских хромосом образуются два потом-

ка. Например, на рис. 4.21, *а*, *б* приведены две родительские хромосомы. Случайным образом определяются точки разрыва. В первой хромосоме — между вершинами $(-, *)$, а во второй — между $(*, 11)$. После реализации стандартного оператора кроссинговера получены два потомка (рис. 4.21, *в*, *г*). При реализации на языке `LISP` операторы кроссинговера сводится к обмену списками между двумя программами при сохранении синтаксической корректности вновь получаемых программ.

4. *Проверка критерия остановки*. Процедура генетического программирования является итерационной, и критерии ее остановки аналогичны критериям для обычных генетических алгоритмов.

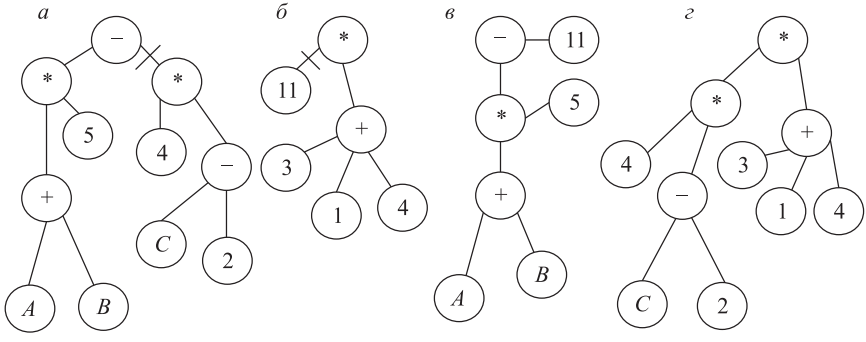


Рис. 4.21. Реализация оператора кроссинговера

5. Рассмотрим перспективные направления исследований в области генетического программирования. К ним относятся работы по так называемым автоматически определяемым функциям, идея которых состоит в повышении эффективности генетического программирования за счет модульного построения программ, состоящих из главной программы и модулей, генерируемых в ходе моделирования эволюции, при этом до начала эволюции ориентировочно определяется архитектура программы, число модулей и параметры (аргументы). Настройка модулей (их число, аргументы и т. п.) зависит от решаемой задачи, имеющихся вычислительных ресурсов и предварительного опыта. Отметим также необходимость типизации вершин дерева, иначе при выполнении оператора кроссинговера могут быть синтезированы синтаксически некорректные решения. Это связано с применением в генетическом программировании модифицированной формы оператора кроссинговера, исключающего появление некорректных решений.

Закон Оккама в приложении к генетическому программированию формулируется так: при анализе нескольких корректных решений лучшим считается решение, обладающее наименьшей сложностью

Другой важный показатель — вычислительная сложность программы. Другим перспективным направлением в генетическом программировании считается вопрос о применении в качестве оператора поиска не только оператора кроссинговера, но и оператора мутации, а также реализация генетического программирования на транспьютерных вычислительных системах.

Решения различной длины, получаемые с помощью генетического программирования, образуют так называемый эффект «компрессионного давления», которым обладают решения с малой структурной сложностью. Программы, генерируемые по методу генетического программирования, могут содержать «лишние» блоки, не влияющие на функциональные возможности программы и на значение ее целевой функции. Принимая во внимание фундаментальную теорему генетических алгоритмов, приведем следующую последовательность рассуждений. Обозначим через $C_{\mathrm{e}}(a_i)$ эффективную сложность программы a_i $(i = 1, 2, \ldots, N)$, $C_{\mathrm{a}}(a_i)$ — абсолютную сложность программы a_i, $P_s(\mathrm{OK})$ — вероятность выживания решения после применения оператора кроссинговера, $P_{\mathrm{d}}(\mathrm{OK})$ — вероятность того, что применение оператора кроссинговера приведет к деструктивному эффекту. Пусть $F(a_i)$ — значение целевой функции программы a_i, $\overline{F(t)}$ — среднее значение целевой функции всех программ в популяции P^t. Если применяется пропорциональная селекция, то нижняя оценка $A_i(t+1)$ «доли» программы a_i в популяции $P^{(t+1)}$ примерно равна

$$A_i(t+1) \cong A_i(t) \cdot F(a_i)/\overline{F(t)} \cdot (1 - P_{\mathrm{c}} \cdot C_{\mathrm{e}}(a_i)/(C_{\mathrm{a}}(a_i) \cdot P_{\mathrm{d}}) = \\ = (F(a_i) - P_{\mathrm{c}} \cdot F(a_i) \cdot C_{\mathrm{e}}(a_i)/C_a(a_i) \cdot P_{\mathrm{d}})A_i(t)/\overline{F(t)}.$$

Выражение в скобках представляет собой определение эффективного значения целевой функции

$$F_{\mathrm{e}}(a_i) = F(a_i) - P_{\mathrm{c}} \cdot F(a_i) \cdot C_{\mathrm{e}}(a_i)/C_{\mathrm{a}}(a_i) \cdot P_{\mathrm{d}},$$

которая соответствует вкладу программы a_i в следующую популяцию $P^{(t+1)}$. Отсюда следует, что репродуктивные шансы некоторой программы a_i тем выше, чем меньше отношение между эффективной и абсолютной сложностью программы. Достигнуть это можно двумя путями.

Первый заключается в увеличении абсолютной сложности, второй — в поиске простых решений. Полученные эмпирические данные подтверждают приведенные выше соображения:

- на ранних этапах эволюции среднее значение целевой функции от популяции к популяции изменяется сильно, в то время как F_{e} изменяется относительно медленно, а репродуктивные свойства некоторой программы зависят, прежде всего, от значения ее целевой функции;

- затем темп изменения целевой функции уменьшается, однако начинает расти соотношение между эффективной и абсолютной сложностью, «компрессионное давление» возрастает и F_{e} уменьшается;
- на заключительных этапах эволюции экспоненциально растет абсолютная сложность программы, F_e остается на относительно низком уровне, среднее значение продолжает улучшаться, а деструктивное влияние оператора кроссинговера уменьшается.

Теоретически обоснованное и эмпирически наблюдаемое явление «компрессии» ведет к преждевременной сходимости генетического программирования к квазиоптимальным решениям. Предлагается ввести внешнее управление этим процессом с помощью некоторого коэффициента D, пропорционального абсолютной сложности программы. С учетом этого выражение для F_e принимает вид

$$F_{\mathrm{e}}(a_i) = F(a_i) - D \cdot C_{\mathrm{a}}(a_i) - P_{\mathrm{c}} \cdot F(a_i) \cdot C_{\mathrm{e}}(a_i)/C_{\mathrm{a}}(a_i) \cdot P_{\mathrm{d}}.$$

Применение фундаментальной теоремы генетического алгоритма для описания динамики эволюционного процесса является упрощенным подходом, поэтому исследуются альтернативные подходы, которые базируются на анализе статистических и динамических особенностей различных форм представления хромосом в популяции.

4.4. Новые структуры генетических операторов

1. Генетические алгоритмы манипулируют популяцией хромосом на основе механизма натуральной эволюции.

По количественному составу хромосом альтернативные решения можно разделить на две группы:

- *монохромосомные* — содержащие только одну последовательность генов (хромосом). Заметим, что в большинстве задач проектирования используются монохромосомные модели;
- *мультихромосомные* — содержащие две и более хромосомы.

По качественному составу мультихромосомные особи можно разделить на следующие два семейства:

- *гомоморфные* — содержащие морфологически и генетически сходные хромосомы. Иначе говоря, все хромосомы данной особи относятся к одному типу, содержат одинаковое число генов, и одинаковые диапазоны допустимых значений генов (аллелей) в соответствующих локусах каждой хромосомы;
- *негомоморфные* — содержащие хромосомы разного типа, различающиеся числом генов или набором допустимых аллелей.

Определим типы хромосом, которые реализуются в подсистемах эволюционного и генетического поиска. По методам представления генов хромосомы можно условно разделить на три типа [1–3].

Двоичные хромосомы — это такие хромосомы, гены которых могут принимать только два значения {0; 1}. Примером двоичной хромосомы является следующая последовательность:

P_0:	1	0	0	1	0	1	1

Для подробного представления таких последовательностей иногда вводится описательная форма. В подсистеме генетического поиска двоичным хромосомам соответствует тип данных GABinaryChromosome, в котором используется побитовая упаковка, т. е. на каждый ген отводится один бит. В некоторых случаях используется три значения: $\{0; 1, *\}$. Здесь $*$ определяется как символ «не имеет значения» 1 или 0.

Числовые хромосомы — это такие хромосомы, гены которых могут принимать целочисленные значения в заданном интервале. В зависимости от диапазона допустимых значений генов (аллелей) можно предусмотреть возможность использования разных типов целых чисел для представления генов: однобайтовое целое (8 бит), позволяющее представить диапазон чисел $[0,\ 255]$, или $[-128,\ +127]$; двухбайтовое целое (16 бит), позволяющее представить диапазон чисел $[0,\ 65\ 535]$, или $[-32\ 768,\ +32\ 767]$; четырехбайтовое целое (32 бита), позволяющее представить диапазон чисел $[0,\ 4\ 294\ 967\ 295]$, или $[-2\ 147\ 483\ 648, +2\ 147\ 483\ 647]$. Такое представление позволяет существенно сократить затраты памяти и, как следствие, повысить эффективность реализуемых алгоритмов.

Среди числовых хромосом выделяют два типа.

Гомологичные хромосомы — это хромосомы, морфологически и генетически сходные, и поэтому не образующие недопустимых решений при применении стандартных генетических операторов. В гомологичных числовых хромосомах каждый ген может принимать целые значения в заданном числовом интервале, при этом для различных генов могут быть заданы различные интервалы $[a_i,\ b_i]$, где $i = \overline{1, n}$, n — число генов в хромосоме. Область поиска решений, которая кодируется такими хромосомами, представляет собой некоторую прямоугольную подобласть в n-мерном пространстве, где R^n — многомерный куб. Примером гомологичной хромосомы является следующая последовательность:

P_1:	1	8	3	0	3	4	1

Негомологичные хромосомы — это такие хромосомы, гены которых могут принимать значения в заданном интервале. При этом интервал одинаков для всех генов, но в хромосоме не может быть двух генов с одинаковым значением. Для негомологичных хромосом применяют различные специальные генетические операторы, не создающие недо-

пустимых решений. Примером негомологичной хромосомы является следующая последовательность:

P_2:	1	4	3	6	5	7	2

Векторные хромосомы — это такие хромосомы, гены которых представляют собой векторы целых чисел. При этом аллели генов обладают свойствами негомологичной числовой хромосомы, т. е. элементы вектора могут принимать значения в заданном интервале, и вектор не может содержать двух одинаковых чисел. Тем не менее, хотя гены в векторных хромосомах негомологичны, сами хромосомы являются гомологичными им. Например,

P_3:	$\langle 1,3,2\rangle$	$\langle 4,5,6\rangle$	$\langle 1,8,6\rangle$	$\langle 7,6,3\rangle$	$\langle 11,5,13\rangle$	$\langle 12,4,8\rangle$	$\langle 9,6,4\rangle$

Таким образом, при разработке подсистемы генетического поиска необходимо реализовать поддержку мультихромосомного представления альтернативных решений задачи, обеспечивая при этом возможность комбинирования двоичных, гомологичных и негомологичных числовых, а также векторных хромосом в одной особи. Простейшим подходом является использование единственного типа хромосом — числовых, и сведение остальных типов к заданному. Очевидно, что простые генетические операторы не всегда применимы к тому или иному типу хромосом. Например, стандартный одноточечный кроссинговер нельзя использовать с негомологичными хромосомами. Невозможность контроля соответствия типов хромосом повлечет возникновение ошибок, связанных с неверным выбором генетического оператора (ГО). Для локализации источника ошибок (неверного оператора) потребуется проверка каждого шага алгоритма. Например, двоичная хромосома может быть представлена битовым массивом, что обеспечит сжатие данных в 32 раза по сравнению с унифицированной числовой хромосомой (в 32-х разрядных операционных системах).

Реализация системы на базе универсального числового типа хромосом позволяет всегда получать допустимые решения. При этом происходит увеличение затрат времени. Такая модель данных обеспечивает возможность сформировать особь с произвольной морфологической и генетической структурой.

2. Опишем варианты представления различных типов данных, их кодирование и декодирование. В задачах оптимизации альтернативное решение часто представляется в виде последовательности неповторяющихся чисел. Такое решение называется *перестановкой*.

Например, пусть необходимо закодировать следующую последовательность чисел: $(1,2,5,8,9)$. Она может моделировать последовательность обхода вершин графа или порядок размещения элементов схемы на коммутационном поле. При прямом кодировании элементов исполь-

зуется числовая негомологичная хромосома, в которой локус — позиция элемента (вершины), аллель — значение элемента в данной позиции. При кодировании с последующим упорядочиванием используется числовая гомологичная хромосома, в которой локус — элемент последовательности, аллель — некоторое значение, в соответствии с которым элементы упорядочиваются при декодировании. При этом диапазон допустимых значений аллелей может быть больше числа элементов последовательности. Например, при упорядочивании по возрастанию последовательность может быть закодирована следующим образом:

1	8	5	2	9

Процесс декодирования данной хромосомы состоит в следующем: хромосома анализируется и определяется взаимно однозначное соответствие номера локуса и значения гена (аллели): $1 \leftrightarrow 1$; $2 \leftrightarrow 8$; $3 \leftrightarrow 5$; $4 \leftrightarrow 2$; $5 \leftrightarrow 9$. Аллели упорядочивается по возрастанию, а затем заменяются на соответствующие им локусы:

Хромосома:	1	8	5	2	9
Упорядоченная последовательность аллелей:	1	2	5	8	9
Декодированная последовательность:	1	4	3	2	5

При упорядочивании по убыванию та же последовательность может быть представлена следующей хромосомой:

9	8	5	2	1

Следует отметить, что подобный способ кодирования не устанавливает взаимнооднозначного соответствия фенотипа и генотипа особи, т. е. одна и та же последовательность может быть представлена несколькими разными хромосомами, равно как и одна хромосома может быть декодирована несколькими разными способами в случае совпадения значений разных генов.

При кодировании методом перестановок используется гомологичная числовая хромосома, гены которой принимают целочисленные значения в диапазоне $[0, n]$, где n — число элементов в кодируемой последовательности. Для декодирования хромосомы используется опорный вектор, содержащий все элементы кодируемой последовательности. Процесс декодирования состоит в последовательности перестановок элементов опорного вектора и имеет линейную временную сложность $O(n)$. При этом номер локуса и аллель (значение гена) задают номера элементов последовательности, которые меняются местами. Следует отметить, что для декодирования всех хромосом должен использоваться один и тот же опорный вектор.

Например, при использовании в качестве опорного вектора $\langle 1, 2, 3, 4, 5\rangle$, приведенная последовательность может быть закодирована следующим образом:

3	5	1	5	2

В процессе анализа данной хромосомы получим следующий порядок перестановок: $1 \leftrightarrow 3$, $2 \leftrightarrow 5$, $3 \leftrightarrow 1$, $4 \leftrightarrow 5$, $5 \leftrightarrow 2$, применяя который к опорному вектору, получим следующую последовательность преобразований:

опорный вектор:	1	2	3	4	5
перестановка:	результат перестановки:				
$1 \leftrightarrow 3$	3	2	1	4	5
$2 \leftrightarrow 5$	3	5	1	4	2
$3 \leftrightarrow 1$	1	5	3	4	2
$4 \leftrightarrow 5$	1	5	3	2	4
$5 \leftrightarrow 2$	1	4	3	2	5

В результате последней перестановки $5 \leftrightarrow 2$ получаем закодированную последовательность $\langle 1, 4, 3, 2, 5\rangle$.

Отличительной особенностью данного способа представления является то, что закодированное решение не зависит от алфавита исходной последовательности. Иначе говоря, элементы опорного вектора могут быть произвольными, покрывать непрерывный или фрагментарный диапазон.

3. Рассмотрим кодирование множества чисел, допускающее многократное вхождение одинаковых элементов. На практике распространены задачи, решение которых представляет собой набор некоторых параметров. При этом области допустимых значений указанных параметров могут перекрываться или совпадать. Примером такой задачи является аппроксимация сложной функции полиномом, решение которой — набор коэффициентов перед слагаемыми соответствующего полинома. Допустим, что необходимо закодировать следующее множество с повторениями:

1	1	2	4	6	1	2	5	4

Рассмотрим кодирование без учета порядка элементов множества. Данный подход предполагает, что порядок следования элементов множества не учитывается. При этом кодируемая информация — это количество повторений того или иного элемента. При таком подходе можно использовать *числовую гомологичную хромосому*, в которой

длина хромосомы совпадает с числом допустимых элементов, локус — элемент множества, аллель — число вхождений данного элемента в кодируемое множество. Заметим, что сумма аллелей хромосомы должна совпадать с мощностью кодируемого множества. Если набор элементов составляют целые числа из диапазона [1, 9], то приведенное выше множество будет кодироваться следующей хромосомой:

3	2	0	2	1	1	0	0	0

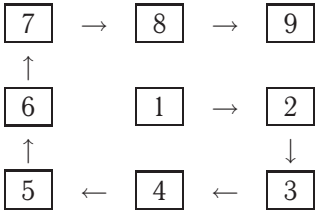
При прямом кодировании множества (с учетом порядка элементов) используется числовая гомологичная хромосома, в которой локус — позиция элемента в множестве, аллель — элемент из допустимого диапазона в данной позиции. Тогда получим

1	1	2	4	6	1	2	5	4

При кодировании многомерной последовательности чисел анализируются задачи, описанные в виде n-мерной матрицы. Рассмотрим подходы к кодированию многомерной последовательности на примере плоской (двумерной) матрицы. Пусть мы имеем следующую двумерную матрицу:

1	6	3
4	8	2
5	9	7

Рассмотрим линеаризацию n-мерного пространства (преобразование в одномерную последовательность). Основная идея состоит в том, чтобы пронумеровать все ячейки в соответствии с ранее заданным правилом (по строкам, по столбцам, вдоль главной/побочной диагонали, по спирали и т. п.). Например, нумерация позиций по правой спирали от центра к краям имеет вид:



Предложенный подход позволяет развернуть приведенное выше размещение (двумерную последовательность) в одномерную:

8	2	7	9	5	4	1	6	3

После подобного преобразования можно использовать любой метод кодирования одномерной последовательности, предложенный выше.

При разложении по векторам подгруппы данных многомерной последовательности могут быть логически объедены. В этом случае такие логически связанные группы данных удобно представить как единичный элемент хромосомы — ген. При таком подходе многомерная последовательность кодируется с использование векторной хромосомы. В ней локус соответствует логически связанной группе данных (строка, столбец матрицы; плоскость и т. п.), а аллель — кортежу $\langle x_1; x_2; \ldots; x_n \rangle$, соответствующему кодируемому подмножеству данных. Разложение по строкам приведенной выше матрицы позволит получить следующую хромосому:

$\langle 1; 6; 3 \rangle$	$\langle 4; 8; 2 \rangle$	$\langle 5; 9; 7 \rangle$

При координатном представлении элементы, размещаемые в n-мерном пространстве, не повторяются. Предлагается использовать векторную хромосому, в которой локус соответствует номеру элемента, а аллель — кортежу вида $\langle x_1; x_2; \ldots; x_n \rangle$ — определяющему «координаты» элемента. Тогда получим

$\langle 1; 3 \rangle$	$\langle 3; 2 \rangle$	$\langle 3; 3 \rangle$	$\langle 1; 2 \rangle$	$\langle 1; 1 \rangle$	$\langle 2; 3 \rangle$	$\langle 3; 1 \rangle$	$\langle 2; 2 \rangle$	$\langle 2; 1 \rangle$

Существует класс задач, решение которых представляется действительными числами или их последовательностью. Например, моделирование функции, имеющей поверхность сложной формы, или подбор коэффициентов связей нейронной сети. Основная сложность при решении таких задач заключается в том, что существующие генетические операторы ориентированны на данные, имеющие дискретную область допустимых значений, тогда как действительные числа определены на непрерывном интервале.

Например, пусть необходимо закодировать действительное число d, определенное на интервале $[a, b]$, с заданное точностью Δ. Здесь используется дискретизация диапазона допустимых значений. Данный подход состоит в том, что диапазон допустимых значений $[a, b]$ разбивается на отрезки длиной Δ, которые нумеруются, начиная с нуля. Обозначим за k — номер отрезка, в который попадает число d, $k_{\max}$ — максимальное число отрезков:

$$k_{\max} = \frac{b-a}{\Delta},$$

тогда преобразование $d \to k$ будет производиться по следующей формуле:

$$k = INT\left(\frac{d-a}{\Delta}\right).$$

При декодировании $k \to d$ возможно несколько вариантов:
— d соответствует левый (правый) конец отрезка k;
— d соответствует середине отрезка k;
— d генерируется случайным образом в диапазоне $[k \cdot \Delta; (k+1) \cdot \Delta]$;

При разложении в дробь действительное число d представляется как отношение двух целых чисел A, R:

$$d \to \frac{A}{R}, \quad \text{при этом} \quad \left| d - \frac{A}{R} \right| \leqslant \frac{\Delta}{2}.$$

Например, $d = 46{,}168$ может быть представлено в виде $\langle 5771; 125 \rangle$.

4. Иногда используется представление в виде числа с плавающей запятой. Данный подход состоит в том, что действительное число d записывается в виде

$$d = m \cdot N^p,$$

где m — мантисса, N — основание, p — порядок. Таким образом, действительное число может быть представлено парой чисел $\langle m, p \rangle$. Например, $d = 46{,}168$ при основании $N = 10$ будет представлено в виде $46168 \cdot 10^{-3}$. Тогда оно кодируется парой чисел $\langle 46168; -3 \rangle$. Отметим, что существует большое число различных методов кодирования информации. Благодаря открытой архитектуре подсистемы генетического поиска пользователь может определить свои уникальные методы кодирования и использовать их в проектируемом генетическом алгоритме.

5. При реализации генетических операторов (ГО) одним из важнейших вопросов является определение места и количества точек разрыва хромосом (альтернативных решений). Большое число точек разрыва может привести к полной потере лучших решений. Малое число точек разрыва часто приводит к попаданию решения в локальный оптимум, далекий от глобального. Поэтому необходим поиск разумного компромисса в этом вопросе.

Предлагается использовать ГО на основе чисел Фибоначчи, золотого сечения, множества Кантора, дихотомического деления и др. Отметим, что необходимо проводить эксперименты и подбирать параметры, управляющие ГО на основе адаптационных процессов.

Рассмотрим построение модифицированных ГО на основе простых чисел и чисел Каталана. Как известно, простые числа легко определяются на основе решета Эратосфена. Приведем, например, ряд простых чисел (ПЧ) не превышающих 100:

1, 3, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97,....

Очевидно, что такой ряд можно продолжить. Предлагается длину ряда выбирать такую, чтобы последнее число ряда ПЧ было меньше, чем $L - 1$. Здесь L — длина хромосомы.

Рассмотрим, например, реализацию различных генетических операторов на основе простых чисел.

6. *Оператор кроссинговера.*

Даны две родительские хромосомы:

P_1 1 | 2 3 | 4 5 6 7 | 8 9 10

P_2 a | b c | d e f g | h I k |

Можно выбрать любое число из ряда длины $L-1=10-1=9$, т. е. например 1, 3 или 7. Известно, что можно выполнить ОК одно, двух или многоточечный. Выберем одновременно три точки разрыва. Здесь можно предложить большое число вариантов получения потомков.

• ***Чередующийся ОК.***

Тогда хромосомы-потомки, получаются путем чередования строительных блоков из P_1 и P_2:

$P_1' = 1$ b c 4 5 6 7 h i k

$P_1' = a$ 2 3 d e f g 8 9 10

На рис. 4.22 показан блок ОК на основе простых чисел.

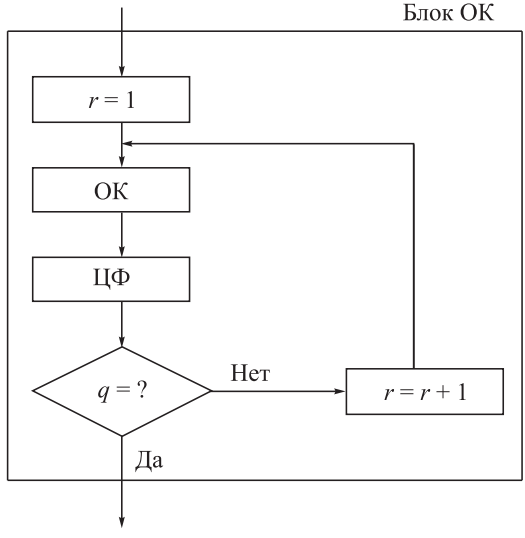


Рис. 4.22. Блок оператора кроссинговера

Здесь r — число точек разрыва; L — длина хромосомы; $r \leqslant L-1$; q — проверка окончания работы блока: она может выполняться по значению ЦФ, по времени, по числу генераций и т. п.

Если в результате ОК получаются нереальные решения, то по определенным правилам происходит замена повторяющегося гена на отсутствующий из хромосомы потомка.

• ***ОК с разным числом точек разрыва.***

Даны две родительские хромосомы:

$P_1 = 1$ 2 3 4 5 6 7 | 8 9 10

$P_2 = 3$ | 5 7 | 9 1 2 4 6 10 8

Число точек разрыва равно *трем* и это согласно ряду ПЧ 1, 3, 7

$P_1' = \overline{\underline{1}\ 5\ 7\ 9\ \underline{1}\ 2\ 4\ \underline{8}\ 9\ \underline{8}}$.

Решение получилось нереальным, так как гены 1 и 8 повторяются дважды, а гены 3 и 10 отсутствуют. В этом случае повторяющиеся гены заменяются на отсутствующие. Тогда получим

$P_1' = 1\ 5\ 7\ 9\ \cancel{1}\ 2\ 4\ \cancel{8}\ 9\ 8 \rightarrow 1\ 5\ 7\ 9\ 3\ 2\ 4\ 8\ 9\ 10$

Аналогично строятся остальные ГО на основе простых чисел (ПЧ).

Рассмотрим реализацию ГО на основе чисел Каталана. По аналогии с числами Фибоначчи существует последовательность чисел Каталана. Она задается формулой

$$k_{\text{am}}(n) = \frac{(2n)!}{(n+1)!\,(n!)}\,.$$

Если последовательность начинается с 1 члена ($n = 1$), то первые 9 элементов определяются так:

$$k_{\text{am}} = 1,\ 2,\ 5,\ 14,\ 42,\ 429,\ 1430,\ 4862.$$

Очевидно, что

$$k_{\text{am}}(0) = 1, \quad \text{так как} \quad 0! = 1 \quad \text{и} \quad k_{\text{am}} = \frac{(2n)!}{(n+1)!\,(n!)} = \frac{(2\cdot 1)!}{2!\,1!} = 1.$$

Тогда последовательность чисел Каталана, начиная с нулевого и до восьмого члена, запишется так:

$$k_{\text{am}} = 1,\ 1,\ 2,\ 5,\ 14,\ 42,\ 429,\ 1430,\ 4862, \ldots\ .$$

Например, при $n = 4$ получим $k_{\text{am}} = \dfrac{(2\cdot 4)!}{(4+1)!\,(4!)} = \dfrac{1\,2\,3\,4\,5 \quad 6\,7\,8}{1\,2\,3\,4\,5 \quad 1\,2\,3\,4} = 14.$

Отметим, что число способов разбиения выпуклого $(n+2)$-угольника на треугольники равно двум. При разбиении выпуклого пятиугольника на треугольники получим 5 способов разбиения, что равно $k_{\text{am}}(3) = 5$.

Выполнение ГА на основе чисел Каталана эффективно при наличии хромосом большой длины при малом числе испытаний. Механизм реализации ГА на основе ПЧ и чисел Каталана идентичны.

7. Рассмотрим ***оператор мутации*** (ОМ). Пусть задана родительская хромосома

$P_1 = 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12$

Здесь также $r = L - 1$. Тогда из последовательности чисел Каталана (ЧК) получим, что можно выполнить разрывы после 1, 2 и/или 5 гена.

Выполним трехточечный ОМ:

$P_1 = 1\,|\,2\,|\,3\,4\,5\,|\,6\,7\,8\,9\,10\,11\,12$

Здесь возможна следующая модификация ОМ:

$P_1 = 2\,3\,1\,4\,6\,5\,7\,8\,9\,10\,11\,12.$

Здесь ген 1 меняется местами с геном 2. Затем ген 1 меняется местами с геном 3 и ген 5 меняется местами с геном 6.

8. Опишем методы построения генетических операторов для решения оптимизационных задач, используя теорию фигурных чисел.

Фигурные числа — общее название чисел, геометрическое представление которых связано с той или иной геометрической фигурой. Понятие восходит к пифагорейцам. Различают следующие виды фигурных чисел.

Линейные числа — числа, не разлагающиеся на сомножители, т. е. их ряд совпадает с рядом простых чисел, дополненным единицей: 1, 2, 3, 5, 7, 11, 13, 17, 19, 23,....

Плоские числа — числа, представимые в виде произведения двух сомножителей: 4, 6, 8, 9, 10, 12, 14, 15,....

Телесные числа — числа, выражаемые произведением трех сомножителей: 8, 12, 18, 20, 24, 27, 28,... .

Многоугольные числа. Если выложить из одинаковых кружков правильный многоугольник, то количество требуемых для этого кружков называется многоугольным числом.

Опишем модифицированные генетические операторы на основе треугольных чисел. Треугольные числа относятся к многоугольным, т. е. треугольное число — это число кружков, которые могут быть расставлены в форме равностороннего треугольника, как показано на рис. 4.23.

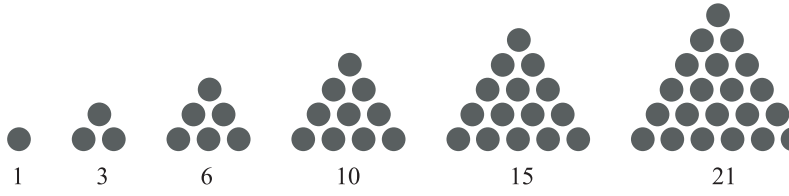


Рис. 4.23. Последовательность треугольных чисел

Последовательность треугольных чисел T_n задается следующей формулой:

$$T_n = \frac{1}{2}n(n+1), \qquad n = 1,\ 2,\ 3,\ \ldots\ .$$

Фигурные числа, особенно треугольные, пользовались большой популярностью при изучении чисел в конце эпохи Возрождения, после того как греческая теория чисел проникла в Западную Европу.

Длина последовательности фигурных чисел выбирается таким образом, чтобы последнее фигурное число было меньше, чем длина хромосомы L. Если длина хромосомы равна 10, то последнее фигурное число для выполнения ГО равно 6. Число точек разрыва выбирается

случайно на основе полученного множества фигурных чисел. Для примера, число точек разрыва может быть 1, 3 или 6. После определения точек разрыва хромосомы-потомки получаются путем чередования строительных блоков родителей. В случае возникновения нереальных решений (при появлении дублирующихся генов) применяется механизм коррекции ошибок на основе взаимного соответствия родительских генов. На основе правил замены генов, полученных после установления соответствия, выполняется замена повторяющихся генов на последних позициях.

Пусть для применения ГО выбраны хромосомы, показанные на рис. 4.24, *а*. При $L = 10$ ряд фигурных чисел для выполнения ГО имеет вид: 1, 3, 6. Случайным образом выбираем число точек разрыва на основе полученного ряда. Выберем три точки разрыва, как показано на рис. 4.24, *б*. Хромосомы-потомки, полученные с помощью алгоритма модифицированного ОК, показаны на рис. 4.24, *в*.

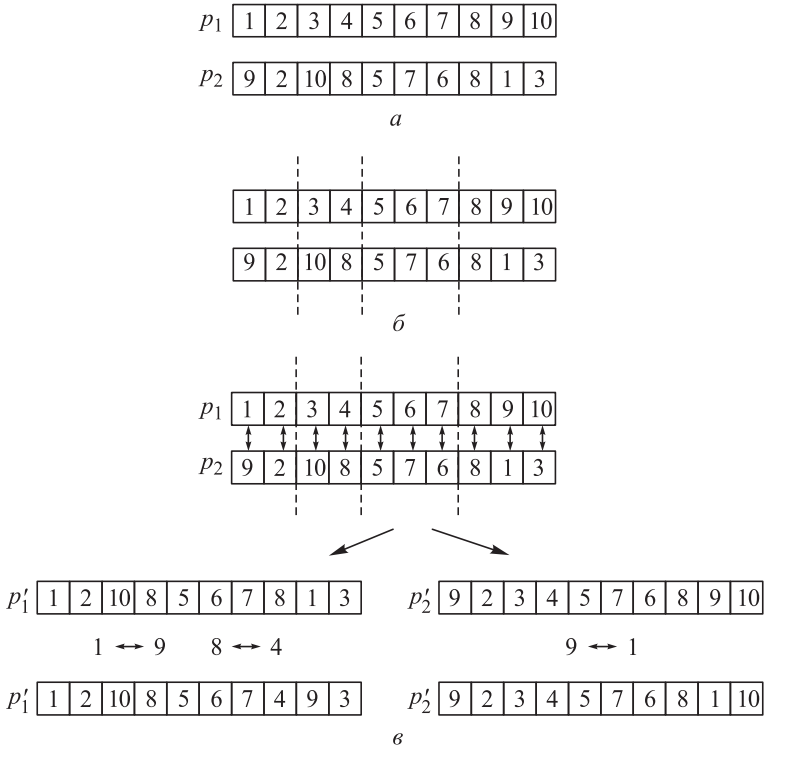


Рис. 4.24. Выполнение ОК на основе фигурных чисел: *а* — хромосомы для применения ГО; *б* — точки кроссинговера в родительских хромосомах; *в* — формирование хромосом-потомков

Вероятность выживания альтернативного решения с лучшим значением ЦФ после первого шага оператора вычисляется по формуле:

$$\Pr(s)[\text{ОКФЧ}] = 1 - \frac{2}{L-1},$$

где L — длина хромосомы. Тогда, соответственно, вероятность устранения альтернативного решения с лучшим значением ЦФ вычисляется по формуле:

$$\Pr_r(d)[\text{ОКФЧ}] = 1 - \Pr_r(s).$$

В рассматриваемом примере имеем: $\Pr_r(s)[\text{ОКФЧ}] = 1 - 2/9 = 7/9$, а $\Pr_r(d)[\text{ОКФЧ}] = 2/9$.

Рассмотрим оператор мутации на основе фигурных чисел. Также как и для ОК, длина последовательности фигурных чисел определяется длиной хромосомы. Однако число точек мутации задается не самой последовательностью, а ее длиной. Например, если длина хромосомы равна 10, то длина ряда равна 3. Поэтому число точек мутации определяется анализируя отрезок $[1;3]$. После определения точек мутации происходит перестановка генов.

Например, пусть имеется родительская хромосома длины 13, показанная на рис. 4.25, *а*. При $L = 13$ ряд фигурных чисел для выполнения ГО имеет вид: 1, 3, 6, 10. Случайным образом выбираем число точек мутации из отрезка $[1;4]$. Выберем 4 точки мутации. Тогда ОМ будут подвержены гены в позициях 1, 3, 6, 10 согласно ряду фигурных чисел (рис. 4.25, *б*). Хромосома-потомок образуется путем перестановки генов между точек мутации, т. е. $1 \leftrightarrow 3$, $13 \leftrightarrow 12$, $11 \leftrightarrow 2$, $\leftrightarrow 5$. Результат выполнения модифицированного оператора мутации показан на рис. 4.25, *в*. Вероятность выживания альтернативного решения с лучшим значением ЦФ после реализации оператора мутации на основе фигурных чисел определяется, как показано выше. Отметим, что остальные модифицированные генетические операторы на основе рассмотренных методов строятся аналогично.

Приведем формальное определение ГА:

$$\text{ГА} = (P_i^0, N, P_{i,k}^T, T, L_j, A, (\text{ЦФ,ОГР,ГУ}), \text{ГО}, t),$$

где P_i^0 — исходная популяция хромосом (альтернативных решений), $P_i^0 = (P_{i1}^0, P_{i2}^0, \ldots, P_{in}^0)$, $P_{i1}^0 \in P_i^0$ — хромосома (альтернативное решение), принадлежащее i-й исходной популяции; N — мощность популяции, т. е. число входящих в нее хромосом, $N = |P_i^T|$; $P_{ik}^T \in P_i^T$ — k-я хромосома, принадлежащая i-й популяции, находящейся в T поколении эволюции; $T = 0, 1, 2, \ldots$ — номер поколения, проходящего популяцией во время эволюции. Иногда число поколений связывают с числом генераций генетического алгоритма, обозначаемых буквой G; L_j — длина i-й хромосомы (альтернативного решения), т. е. число генов (элементов, входящих в закодированное решение, представлен-

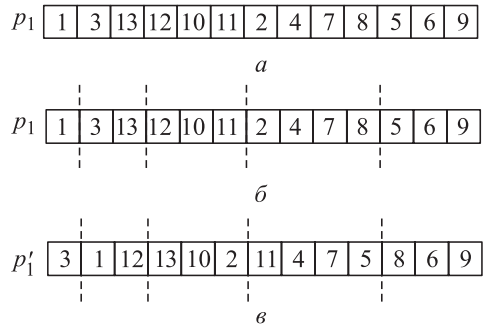


Рис. 4.25. Выполнение ОМ на основе фигурных чисел: *а* — хромосома для применения модифицированного ОМ; *б* — точки мутации; *в* — результат применения ОМ на основе фигурных чисел

ное в заданном алфавите), например, $|P_i^T| = L_j$; A — произвольный абстрактный алфавит, в котором, кодируются хромосомы, например, $A_1 = \{0, 1\}$, $A_2 = \{0, 1, 2, \ldots, 10\}$, $A_3 = \{0, 1, 2, *\}$, $A_4 = \{A, B, C, D\}$; здесь $*$ — метка, означающая любой символ в алфавите A_2; (ЦФ, ОГР, ГУ) — целевая функция, ограничения и граничные условия, которые определяются на основе заданной модели исходной решаемой задачи; ГО — генетические операторы, t — критерий окончания работы ГА.

4.5. Параллельные генетические алгоритмы

1. С развитием возможностей вычислительных средств многоядерных процессоров параллельные вычисления позволяют решать сложные задачи за меньшее время. Параллельный генетический алгоритм (ПГА) использует две главные модификации по сравнению с генетическим алгоритмом. Во-первых, выбор для скрещивания распределен. Особи живут в двумерном мире. Выбор особи для скрещивания производится каждым индивидуумом независимо от его соседства. Во-вторых, каждая особь может улучшать свою целевую функцию в течение всей жизни. Стратегия поиска ПГА основана на небольшом количестве активных и интеллектуальных особей, тогда как ГА использует огромную популяцию пассивных особей.

Распределенные эволюционные алгоритмы — это подкласс параллельных генетических алгоритмов, предназначенных для снижения преждевременной сходимости к локальному оптимуму, стимуляции разнообразия и поиска альтернативных решений. Они основаны на разбиении популяции на несколько отдельных популяций, каждая из которых будет обрабатываться ГА независимо от других. Кроме того, разнообразные миграции индивидов порождают обмен генетическим материалом среди популяций, которые обычно улучшают точность и эффективность алгоритма.

Мощность эволюционных алгоритмов усиливается с применением распределенных вычислений.

Существует три главных типа параллельных ГА:

1) глобальные одно-популяционные ПГА, модель «хозяин–раб» («мастер–подчиненные») (master-slave GAs);
2) однопопуляционные ПГА (fine-grained GAs);
3) многопопуляционные ПГА (coarse-grained GAs).

В алгоритмах первого типа существует главная популяция, но оценка целевой функции (ЦФ) распределена среди нескольких процессоров. Хозяин хранит популяцию, выполняет операции ГА и распределяет индивидуумы между подчиненными. Они оценивают ЦФ индивидуумов. Эти алгоритмы называют также глобально параллельными ГА.

Однопопуляционные ГА пригодны для массовых параллельных компьютеров и состоят из одной пространственно-структурированной популяции. Селекция и скрещивание ограничены близкой родственностью. Этот класс параллельных ГА имеет одну пространственно-распределенную популяцию и он может быть эффективно реализован на массовых параллельных компьютерах.

Многопопуляционные (или многообщинные) ГА более сложные, так как они состоят из нескольких подпопуляций, которые периодически обмениваются индивидуумами. Этот обмен индивидуумами называется *миграцией* и управляется несколькими параметрами.

Многообщинные ГА — популярный параллельный метод. многообщинные параллельные ГА известны под различными именами. Иногда они известны как «распределенные» ГА, потому что они обычно реализуются на MIMD компьютерах с распределенной памятью. Многообщинные ГА имеют сходство с «моделью островов» в популяционной генетике (Population Genetics).

Рассмотрим островную модель параллельных генетических алгоритмов. Главные характеристики многообщинных параллельных ГА — это использование небольшого числа относительно больших подпопуляций и миграций особей между ними.

В параллельном ГА копия лучшего индивидуума, найденного в каждой общине, посылается всем его соседям после каждого поколения. Цель этой неизменной коммуникации состояла в обеспечении хорошего перемешивания индивидуумов. Параллельные ГА с таким высоким уровнем коммуникации находили решения того же качества, что и последовательный ГА с одиночной большой популяцией.

Существует параллельный ГА с общинами, соединенными в топологии $4D$-гиперкуба. В этом алгоритме миграция совершается в фиксированные интервалы между процессами в направлении одного измерения гиперкуба. Мигранты были выбраны вероятностно из лучших индивидуумов в подпопуляции, и они замещали худшие индивидуумы в популяции приемнике.

Отметим, что многообщинные ГА выглядят как простое расширение последовательного ГА. При этом используется следующая схема:

взять несколько простых последовательных ГА, запустить их на узле параллельного компьютера и в предопределенное время обмениваться несколькими индивидуумами.

2. Основные проблемы, решаемые при моделировании параллельных ГА. В зависимости от класса параллельного ГА (ПарГА) разнятся и задачи, решаемые при его разработке. Выделим основные задачи при создании ПарГА:

- выбор или разработка стратегии взаимодействия составных частей алгоритма;
- подбор частоты миграций между популяциями;
- определение мигрируемых особей и их количества;
- определение структуры эволюции отдельных популяций.

Рассмотрим каждую задачу подробней. Топология — это важный фактор в производительности параллельного алгоритма, потому что она определяет, как быстро (или как медленно) хорошее решение распространяется в другие популяции. Если топология имеет частую связность, хорошие решения будут быстро распространяться во все общины и могут быстро поглотить популяцию. С другой стороны, если топология редко связная, решения будут распространяться медленнее и общины будут более изолированными друг от друга, давая возможность появляться различным решениям. Эти решения могут дольше параллельно развиваться и рекомбинировать для получения потенциально лучших решений.

Топология коммуникации также важна, потому что это главный фактор в стоимости миграции. Например, часто связная топология может способствовать лучшему перемешиванию индивидуумов, но она также влечет за собой более высокие стоимости.

Общая тенденция в многообщинных параллельных ГА — это использование статичных топологий, которые указаны в начале запуска алгоритма и остаются неизменными. Большинство реализаций параллельных генетических алгоритмов со статическими топологиями используют природную (естественную) топологию компьютера, доступную исследователям.

Другой метод конструирования топологии — это использование динамической топологии. В этом методе община не ограничена коммуникацией с некоторым фиксированным количеством общин, вместо этого мигранты посылаются общинам, которые удовлетворяют некоторому критерию. Мотивация за динамическими топологиями — это идентифицирование общин, где мигранты, вероятно, произведут некоторый эффект. Обычно критерий, используемый для выбора общины как места назначения, включает меры разнообразия популяции или меру генотипического расстояния между двумя популяциями (или характерного индивидуума популяции, например, лучшего). При такой архитектуре необходим механизм определения событий в соседних популяциях, причем если в одной из соседних популяций событие наступило, а во

второй нет, то необходимо ожидать наступления события и во второй популяции. Или же возможен односторонний обмен хромосомами.

Частота миграций влияет на конечное решение. Как известно, частые миграции приводят к вырождению популяций, а редкие, наоборот, к снижению сходимости. Для регулирования частоты миграции применяются различные методы, которые можно разбить на два типа: адаптивные и событийные. Первый использует методы адаптации для настройки частоты миграции в процессе работы алгоритма. Второй применяет методы определяющие необходимость миграции, т. е. миграция осуществляется только при наступлении какого-либо события.

Выбор особей для миграции заключается в проблеме решаемой механизмом селекции. Ведь отдельные части хромосомы могут содержать заведомо оптимальную часть генетического материала, и эти части могут находиться в хромосомах с плохой приспособленностью. Исключение таких хромосом полностью может привести к преждевременной сходимости или пропуску глобального оптимума.

На рис. 4.26 приведем схему работы буферной модели параллельного генетического алгоритма.

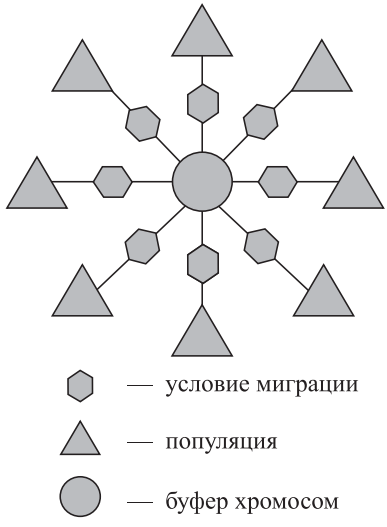


Рис. 4.26. Схема работы буферной модели параллельного генетического

Модель представлена как звездная структура, взаимодействующая с популяциями через так называемый буфер хромосом. Буфер будет заполняться самими популяциями в процессе работы. Весь процесс можно описать следующим образом:

I. Формируются все популяции, и запускаются на выполнение в асинхронном режиме.

II. При наступлении определенной ситуации в популяции эта популяция обращается к буферу и забирает оттуда часть или все хромосомы, затем добавляет туда часть своих особей.

При данной структуре необходимо определить механизм регулирования размера буфера. Достоинство этой модели в большой гибкости, она позволит провести исследования различных стратегий и критериев параллельных генетических алгоритмов.

Приведем примерную структурную схему работы подобного параллельного генетического алгоритма (рис. 4.27). Каждая популяция эволюционирует отдельно от других. На каждой итерации проверяется условие необходимости миграции. Таковым условием может быть интервал итераций, вырожденность популяции и т. п. Если условие

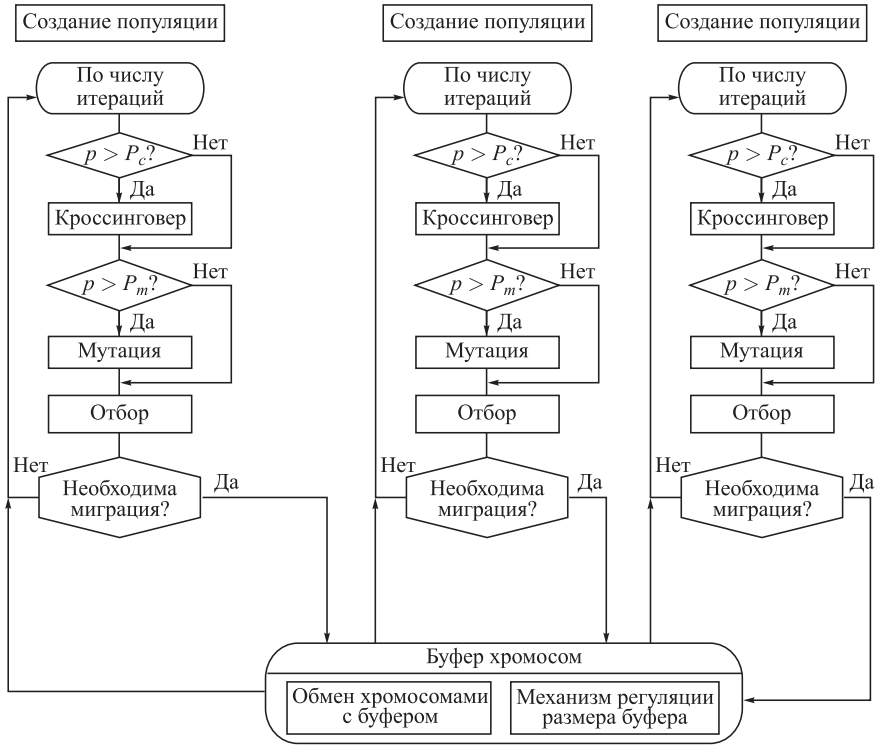


Рис. 4.27. Примерная схема работы параллельного генетического алгоритма с использованием буфера хромосом

наступило, происходит миграция хромосом особей с буфером хромосом. Затем проверяется размер буфера, и если он больше заданного значения, то выполняется процедура отбора. Условия миграции могут быть разными в популяциях. Также можно менять и типы генетических операторов для популяций. Все это позволит провести исследования влияния различных стратегий и механизмов на конечный результат работы ГА.

Отметим, что возможности современных вычислительных устройств можно использовать на полную мощность только в асинхронных моделях. В дальнейшем необходимы исследования внедрения гибридных методов в параллельный ГА на различных этапах функционирования.

4.6. Выводы

Описаны генетические операторы, использующие идеи построения фракталов, методов поиска на основе дихотомического разбиения, чисел Фибоначчи, золотого сечения и др. Преимущество всех этих мето-

дов в том, что при их использовании каждая новая генерация генетических алгоритмов приводит к сокращению интервала неопределенности, т. е. области поиска. Исследователи разрабатывают различные схемы поиска и строительные блоки, на основе которых можно получать алгоритмы различных уровней сложности. В оптимизационных задачах принятия решения находят применение нестандартные архитектуры эволюционного поиска. Для нахождения квазиоптимальных решений за приемлемое время используются модели синтетической эволюции.

Генетические алгоритмы и генетическое программирование являются основными формами эволюционного поиска. Каждая из этих форм имеет свои отличительные черты и особенности. С практической точки зрения необходимо установить, для каких конкретно инженерных задач лучше подходит та или иная форма моделирования эволюции.

4.7. Контрольные вопросы

1. Опишите три различных механизма работы оператора рекомбинации.
2. Приведите принципы работы сканирующего алгоритма.
3. Каким образом можно оценить качество конкретного генетического алгоритма?
4. Приведите гипотезы о размере популяций и числе точек разрыва в генетических операторах.
5. В чем заключается диагональный оператор кроссинговера?
6. Определите понятие фрактального объекта.
7. Дайте определение фрактального множества.
8. Определите способ построения множества Кантора.
9. Приведите способ построения «ковра Серпинского».
10. Приведите пример построения кривой Коха.
11. Каким образом определяется фрактальная размерность рассматриваемого объекта?
12. Что такое самоподобный фрактал?
13. Покажите пример работы оператора кроссинговера множества Кантора (ОКМК).
14. Приведите пример работы оператора мутации множества Кантора (ОКМК).
15. Покажите пример работы оператора кроссинговера дихотомии.
16. Приведите пример работы оператора мутации дихотомии.
17. Приведите пример работы оператора мутации Фибоначчи.
18. Постройте пример работы оператора кроссинговера Фибоначчи.
19. Покажите пример работы оператора инверсии Фибоначчи.
20. Покажите примеры работы операторов сегрегации Фибоначчи.
21. Покажите примеры работы операторов транспозиции и транслокации Фибоначчи.
22. Покажите пример работы оператора инверсии «золотого сечения».

23. Постройте примеры работы операторов кроссинговера и мутации «золотого сечения».
24. Покажите пример работы оператора сегрегации «золотого сечения».
25. Покажите примеры работы операторов транспозиции и транслокации «золотого сечения».
26. Приведите основные методы построения целевой функции.
27. Какие многохромосомные генетические операторы Вы знаете? Приведите примеры их работы.
28. Приведите базовую структуру генетического алгоритма для решения задач оптимизации.
29. Дайте определение микро-, макро- и метаэволюции.
30. Дайте определение эволюционного поиска.
31. В чем заключается процедура миграции? Приведите пример ее работы.
32. Опишите архитектуру эволюционного поиска с миграцией.
33. Приведите схемы эволюционного поиска на основе платоновых графов (тетраэдр; гексаэдр (куб); октаэдр; икосаэдр; додекаэдр).
34. Приведите схемы параллельного и последовательного эволюционного поиска.
35. Приведите схемы эволюционного поиска на основе треугольников и полного графа и ячеечной структуры.
36. Приведите схему реализации процесса метагенетической оптимизации.
37. Постройте реализацию схемы «поиск–эволюция».
38. Постройте реализацию схемы «эволюция–поиск».
39. Постройте реализацию схемы «поиск–эволюция–поиск».
40. Постройте реализацию схемы «эволюция–поиск–эволюция».
41. Постройте реализацию схемы «эволюция–поиск–эволюция–поиск–эволюция–поиск».
42. В чем заключается стохастически-итерационный метод?
43. В чем заключается метод прерывистого равновесия?
44. Опишите работу эволюционного поиска при совместном использовании моделей эволюций Дарвина и Ламарка.
45. Приведите примеры различных схем строительных блоков.
46. Что является задачей принятия решений?
47. Опишите работу комбинированной схемы принятия решения с обратной связью.
48. Опишите работу комбинированной схемы принятия решения на основе блока эволюционной адаптации.
49. Опишите работу базовой структуры генетического поиска на основе использования моделей эволюций Дарвина, Ламарка, де Фриза.
50. Опишите работу модифицированной базовой структуры генетического поиска на основе использования моделей эволюций Дарвина, Ламарка, де Фриза и Поппера.
51. Какие подходы к структурированию популяции вы знаете?

52. Опишите двухуровневую систему генетического алгоритма Гудмана и Панча.
53. Опишите принцип построения параллельных генетических алгоритмов со скрещиванием в локальной области и на основе «островов».
54. Опишите понятие кластерно-ориентированного генетического алгоритма.
55. Каков принцип работы проблемно-ориентированных генетических алгоритмов?
56. Какие варианты планирования процесса генетического поиска вы знаете?
57. Опишите технологию «устойчивой репродукции» Девиса.
58. Как работает групповой генетический алгоритм с направленной мутацией?
59. Приведите схему группового генетического алгоритма с направленной мутацией.
60. Дайте определение генетического программирования.
61. Опишите свойства языка программирования `LISP`.
62. Перечислите стартовые условия для генетического программирования.
63. Опишите методику генетического программирования.
64. Опишите работу оператора кроссинговера в процессе генетического программирования.

4.8. Упражнения

1. Постройте пример реализации сканирующего алгоритма.
2. Постройте алгоритм реализации диагонального оператора кроссинговера.
3. Запишите алгоритм построения множества Кантора.
4. Запишите алгоритм построения «ковра Серпинского».
5. Запишите алгоритм построения «снежинки» (кривой Коха).
6. Постройте алгоритм реализации оператора кроссинговера на основе множества Кантора.
7. Постройте алгоритм реализации оператора мутации на основе множества Кантора.
8. Постройте алгоритм реализации оператора инверсии на основе множества Кантора.
9. Постройте алгоритм реализации оператора сегрегации на основе множества Кантора.
10. Постройте алгоритм реализации оператора транслокации на основе множества Кантора.
11. Постройте алгоритм реализации оператора транспозиции на основе множества Кантора.
12. Постройте алгоритм работы оператора кроссинговера дихотомии.
13. Постройте алгоритм работы оператора мутации дихотомии.

14. Постройте алгоритм работы оператора инверсии дихотомии.
15. Постройте алгоритм работы оператора сегрегации дихотомии.
16. Постройте алгоритм работы оператора транслокации дихотомии.
17. Постройте алгоритм работы оператора транспозиции дихотомии.
18. Постройте алгоритм работы оператора кроссинговера Фибоначчи.
19. Постройте алгоритм работы оператора мутации Фибоначчи.
20. Постройте алгоритм работы оператора инверсии Фибоначчи.
21. Постройте алгоритм работы оператора сегрегации Фибоначчи.
22. Постройте алгоритм работы оператора транслокации Фибоначчи.
23. Постройте алгоритм работы оператора транспозиции Фибоначчи.
24. Постройте алгоритм работы оператора кроссинговера золотого сечения.
25. Постройте алгоритм работы оператора мутации золотого сечения.
26. Постройте алгоритм работы оператора инверсии золотого сечения.
27. Постройте алгоритм работы оператора сегрегации золотого сечения.
28. Постройте алгоритм работы оператора транслокации золотого сечения.
29. Постройте алгоритм работы оператора транспозиции золотого сечения.
30. Запишите алгоритмы построения целевых функций.
31. Постройте псевдокод базового генетического алгоритма.
32. Постройте псевдокод алгоритма эволюционного поиска на основе миграции.
33. Постройте псевдокод алгоритма эволюционного поиска на основе платоновых графов.
34. Постройте псевдокоды алгоритмов последовательного эволюционного поиска и поиска с циклами.
35. Постройте псевдокод алгоритмов эволюционного поиска на основе полных графов.
36. Постройте псевдокод алгоритма эволюционного поиска на основе ячеечной структуры.
37. Постройте псевдокод алгоритма процесса метагенетической оптимизации.
38. Постройте алгоритм реализации стратегии «эволюция–поиск–эволюция–поиск». В качестве эволюции взять модели Ч. Дарвина и К. Поппера, в качестве эвристики поиска — поиск в глубину и метод дихотомии.
39. Постройте псевдокод алгоритма реализации стратегии «эволюция–поиск».
40. Постройте псевдокод стратегии эволюционного поиска при горизонтальной схеме.
41. Постройте псевдокод алгоритма реализации архитектуры соподчинения в эволюционном поиске.
42. Постройте алгоритм стохастически-итерационного метода.
43. Постройте алгоритм метода прерывистого равновесия.

44. Постройте псевдокод совместного алгоритма эволюций.
45. Постройте псевдокод алгоритма генетического поиска на основе использования эволюций Дарвина, Ламарка и де Фриза.
46. Постройте псевдокод алгоритма параллельного эволюционного поиска.
47. Постройте псевдокод алгоритма генетического программирования.

Глоссарий к разделу 4

Временна́я сложность алгоритма — зависимость времени получения результата от количества входных данных.

Генетическое программирование — это направление эволюционного поиска, ориентированное на решение задач автоматического синтеза программ. Хромосомы (компьютерные программы различной величины и сложности) автоматически генерируются с помощью генетических операторов.

Граф — математический объект, состоящий из множества вершин (точек) и множества ребер (линий), находящихся в заданном отношении.

Групповой генетический алгоритм с направленной мутацией. Он состоит из двух уровней. Верхний уровень алгоритма выполняет групповой, а нижний — индивидуальный поиск.

Дерево — граф без циклов.

Задача принятия решений — кортеж, состоящий из двух элементов. Первый элемент — множество вариантов решений задачи, второй — ***принцип оптимальности***, дающий представление о качестве вариантов.

Закон Оккама в приложении к генетическому программированию формулируется так: при анализе нескольких корректных решений лучшим считается решение, обладающее наименьшей сложностью.

Качество — степень, с которой совокупность присущих решению задачи характеристик удовлетворяет заданным требованиям.

Кластерно-ориентированный генетический алгоритм основан на стратегии быстрого разбиения поисковых пространств на области высоких значений целевой функции.

Ковер Серпинского — обобщение множества Кантора на случай плоских фигур.

Лицо, принимающее решение (***ЛПР***) — пользователь, определяющий дальнейшие шаги реализации эволюционного поиска.

Макроэволюция — создание одной популяции и реализация на ней эволюционного поиска.

Метагенетическая оптимизация основана на реализации генетических алгоритмов, генерации новых решений, определении значений целевой функции и использовании предыдущих решений для генерации лучших результатов.

Метаэволюция — создание множества популяций и реализация на нем эволюционного поиска.

Метод прерывистого равновесия использует палеонтологическую теорию, которая строит модели эволюции на основе описаний вулканических и других изменений земной коры.

Миграция — процедура установления порядка обмена хромосом между популяциями.

Микроэволюция — создание одной хромосомы и реализация на ее основе эволюционного поиска.

Многохромосомный механизм глобальной рекомбинации основан на анализе более двух хромосом родителей для построения хромосом потомков.

Множество Кантора (***МК***) — множество, обладающее геометрической инвариантностью (иногда называется «множество средних третей»).

Платоновы графы — правильные многоугольники.

Поиск–эволюция — процедура последовательной оптимизации на основе сначала поискового метода, а затем — генетического алгоритма.

Полный граф — граф, в котором любая пара вершин соединена ребром.

Проблемно-ориентированный генетический алгоритм вырабатывает решения (фенотипы) комбинаторных задач, представленных в виде задач параметрической оптимизации, состоящих из строк параметров (генотипов).

Процедура — установленный порядок выполнения какой-либо деятельности или процесса.

Сканирующий алгоритм базируется на процедуре просмотра популяции, когда хромосома потомок строится слева направо.

Стохастически-итерационный метод основан на определении стартовых точек для направленного поиска и реализации генетических операторов с использованием эволюционной адаптации.

Фрактальное множество — множество, обладающее геометрической (масштабной) универсальностью. Здесь результат одной итерации является начальным условием для другой и требуется нелинейная зависимость между результатом и реальным значением.

Фрактальный объект — самоподобный объект, вид которого не претерпевает существенных изменений при изменении его масштабов.

Эволюционная адаптация — приспособление результатов работы генетических алгоритмов на каждой генерации к условиям внешней среды.

Эволюция–поиск — процедура последовательной оптимизации на основе сначала генетического алгоритма, а затем — поискового метода.

Эффективность — степень реализации запланированной деятельности и достижения запланированных результатов.

Список литературы к разделу 4

1. *Батищев Д. А.* Генетические алгоритмы решения экстремальных задач. — Воронеж: Изд-во ВГТУ, 1995.
2. *Букатова И. Л.* Эволюционное моделирование и его приложения. — М.: Наука, 1994.
3. *Гладков Л. А.*, *Зинченко Л. А.*, *Курейчик В. В. и др.* Методы генетического поиска. — Таганрог: Изд-во ТРТУ, 2002.
4. *Гладков Л. А.*, *Курейчик В. В.*, *Курейчик В. М.* Генетические алгоритмы. Учебное пособие. — Ростов-на-Дону: Ростиздат, 2004.
5. *Дубинин Н. П.* Избранные труды. Т. 1. Проблемы гена и эволюции. — М.: Наука, 2000.
6. *Емельянов В. В.*, *Курейчик В. В.*, *Курейчик В. М.* Теория и практика эволюционного моделирования. — М.: Физматлит, 2003.
7. *Кроновер Р. М.* Фракталы и хаос в динамических системах. Основы теории. — М.: Постмаркет, 2000.
8. *Курейчик В. В.* Эволюционные, синергетические и гомеостатические методы принятия решений. — Таганрог: Изд-во ТРТУ, 2001.
9. *Курейчик В. М.* Генетические алгоритмы и их применение. — Таганрог: Изд-во ТРТУ, 2002.
10. *Люггер Дж.* Искусственный интеллект. Стратегии и методы решения сложных проблем. — М.: Изд. дом «Вильямс», 2003.
11. *Редько В. Г.* Эволюционная кибернетика. — М.: Наука, 2001.
12. *Лахути Д. Г.*, *Садовский В. Н.*, *Финн В. К.* Эволюционная эпистемология и логика социальных наук: Карл Поппер и его критики. — М.: Эдиториал УРСС, 2000.
13. *Koza J. R.* Genetic Programming. — Cambridge/MA: MIT Press, 1994.
14. *Michalewicz Z.* Genetic Algorithms + Data Structures = Evolution Programs. — Berlin: Springer, 1999.

Генетические алгоритмы и генетическое программирование являются основными формами эволюционного поиска. Каждая из этих форм имеет свои отличительные черты и особенности. С практической точки зрения необходимо установить, для каких конкретно инженерных задач лучше подходит та или иная форма моделирования эволюции

5. ОПТИМИЗАЦИОННЫЕ ЗАДАЧИ НА ГРАФАХ

> Новая истина побеждает обычно не так, что ее противников удается переубедить, и они осознают свою неправоту. Дело, попросту говоря, в том, что они вымирают, а подрастающая научная смена сразу усваивает новые истины.
>
> *М. Планк*

5.1. Генетические алгоритмы разбиения графов

1. Одной из важнейших инженерных оптимизационных задач является разбиение графа на заданное или произвольное число частей. Задача разбиения графа на части имеет много практических применений. Она используется при проектировании устройств автоматики и вычислительной техники, создании систем управления, компьютерных и инженерных сетей, а также при решении различных задач искусственного интеллекта. Отметим, что задача разбиения графа относится к классу NP — полных проблем. Следовательно, не существует эффективных алгоритмов ее решения с полиномиальной ***временной сложностью*** (см. приложение 1). ***Временная сложность алгоритма*** (***ВСА***) — это зависимость времени работы алгоритма от числа входов (исходных данных задачи).

В ***полиномиальных алгоритмах*** временная сложность составляет $O(n), O(n^2), O(n^3), \ldots$. Для экспоненциальных алгоритмов (NP) временная сложность алгоритма составляет $O(n!)$ и т. д., $O(n^n), O(n^{2n}), O(n^{3n}), \ldots$. Класс NP-полных задач включает такие задачи, для которых не найдены полиномиальные алгоритмы, однако и не доказано, что их не существует. Для рассмотрения вопроса о NP-полноте оптимизационной задачи их преобразуют в задачу разрешения. Обычно в качестве такой задачи рассматривают проверку, является ли некоторое число верхней или нижней границей для оптимизируемой величины. Если для оптимизационной задачи имеется быстрый алгоритм, то и полученную из нее задачу разрешения можно решать быстро. Для этого надо сравнить ответ этого алгоритма с заданной границей. Говорят, что алгоритм решает оптимизационную

задачу за время $O(T(n))$, если на входных данных длины n алгоритм работает время $O(T(n))$. В этой связи разрабатываются различные эвристики, основанные на идеях последовательных и итерационных алгоритмов.

2. Сформулируем постановку задачи разбиения графа на заданное или произвольное число частей. Пусть задан граф $G = (X, U)$, где X представляет множество вершин графа, U — множество ребер. Пусть $B = \{B_1, B_2, \ldots, B_s\}$ — множество разбиений графа G на части $B_1, B_2, \ldots, B_s$, такие, что $B_1 \cap B_2 \cap \ldots \cap B_s = \varnothing$, и $B_1 \cup B_2 \cup \cup \ldots \cup B_s = B$. Пусть каждое разбиение B_i состоит из элементов $B_i = \{b_1, b_2, \ldots, b_n\}$, $n = |X|$. Тогда задача разбиения графа G на части заключается в получении разбиения $B_i \in B$, удовлетворяющего трем условиям и ограничениям:

$$(\forall B_i \in B)(B_i \neq \varnothing),$$

$$(\forall B_i, B_j \in B)([B_i \neq B_j \to X_i \cap X_j = \varnothing] \wedge \\ \wedge [(U_i \cap U_j = U_{ij}) \vee (U_i \cap U_j = \varnothing)]), \quad (5.1)$$

$$\bigcup_{i=1}^{s} B_i = B, \ \bigcup_{i=1}^{n} U_i = U, \ \bigcup_{i=1}^{n} X_i = X, \ |U_{i,j}| = K_{i,j.}$$

Целевая функция для разбиения графа G запишется так:

$$K = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} K_{i,j}, \quad (i \neq j), \quad (5.2)$$

где $K_{i,j}$ — число связей между частями B_i и B_j при разбиении графа G на части; s — количество частей в разбиении; K — суммарное количество ребер при разбиении графа на части.

Стандартная задача разбиения заключается в минимизации K ($K \to \min$). Минимизация K при разбиении графа на части позволяет косвенно учитывать многие критерии исследуемой модели при решении оптимизационной задачи.

2.1. Рассмотрим задачу разбиения графа G с минимизацией K. Следовательно, задача разбиения состоит в отыскании такого разбиения B_i из множества возможных разбиений B некоторого графа G, при котором минимизируется некоторая величина K, являющаяся целевой функцией разбиения, и учитываются все поставленные в задаче ограничения и граничные условия, если они существуют.

В задачах разбиения графов общее число вариантов решения равно числу перестановок из n вершин графов, т. е. $C_n = n!$, а с учетом ограничений на формирование подмножеств (частей графа в задаче разбиения) — числу сочетаний из n вершин по m частей, т. е.

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

2.2. Из вышесказанного следует, что решение задачи разбиения графа на основе полного перебора затруднительно из-за экспоненциальной сложности процесса. В этой связи разрабатываются различные эвристики решения данных задач. К ним относятся итерационные методы парных и групповых перестановок, методы последовательного приближения, поиска в глубину и ширину, направленного перебора и др. В последнее время появились эвристики, использующие различные методы случайности. Это методы эволюционного моделирования, отжига, генетического поиска и их модификации. Все они в каком-то смысле основаны на анализе и переборе вариантов решения, однако различаются по технологии и принципам реализации.

Переборные методы предназначены для поиска оптимального решения на некотором конечном множестве. В большинстве случаев они используют различные варианты сокращенного перебора, поскольку полный перебор в реальных задачах неосуществим из-за слишком большой размерности. Наиболее распространен для разбиения графов небольшой размерности метод ветвей и границ и его модификации. Временная сложность алгоритмов данного типа лежит в пределах от $O(n^3)$ до $O(n^5)$.

2.3. В основе эвристических методов лежат нечеткие логические рассуждения. Данные методы не гарантируют нахождения оптимального решения и сильно зависят от класса и описания решаемой задачи. Эвристические методы на практике позволяют получать неплохие результаты, однако предсказать их поведение невозможно. Они могут использоваться как самостоятельно, так и совместно с другими. Это позволяет сокращать время работы. Временная сложность алгоритма в них может быть любой, но большинство известных алгоритмов имеют сложность от $O(n)$ до $O(n^2)$.

2.4. Сущность большинства алгоритмов разбиения графов заключается в выборе некоторого начального разбиения исходного графа и последующего его улучшения с помощью итерационного, парного или группового обмена вершин из различных частей разбиения. При этом для каждой итерации осуществляется перестановка таких вершин, которая обеспечивает максимальное уменьшение числа связей между частями разбиения графа или минимальное приращение числа внешних ребер.

3. Приведем краткое описание известных на сегодняшний день алгоритмов, использующих ту или иную эвристическую базу.

Одними из популярных алгоритмов разбиения графа $G = (X, U)$ на части являются итерационные алгоритмы, требующие для своей работы некоторого начального разбиения. Они довольно чувствительны к начальному разбиению, что, в основном, сказывается на времени работы алгоритма. Кроме того, они часто попадают в локальный оптимум, когда размер графа велик ($n > 1000$). Один из путей пре-

одоления этого недостатка состоит в группировке сильно связанных вершин в кластеры и в дальнейшем сборе этих кластеров в заданные части разбиения, а также использование фрактальных множеств для группирования сильно связанных вершин. Итерационный эвристический алгоритм начинается со случайного разбиения и затем применяет парный обмен между двумя подмножествами, улучшая начальное решение путем многократной перестановки вершин. Разработано много модификаций этого подхода для того, чтобы алгоритм мог работать с гиперграфами, уменьшив временную сложность алгоритма до $O(n)$ для графов специального вида.

3.1. Построим схему разбиения графа на части на основе генетического поиска. Такая схема дана на рис. 5.1. Здесь в первом блоке строится текущая (начальная) популяция решений, т. е. определяется заданное подмножество $B_1, B_2, \ldots, B_k$ $(k \leqslant s)$. На построение популяции оказывает влияние внешняя среда в виде ЛПР или ЭС. Отметим, что такое подмножество решений может быть получено случайным, направленным или случайно-направленным методами.

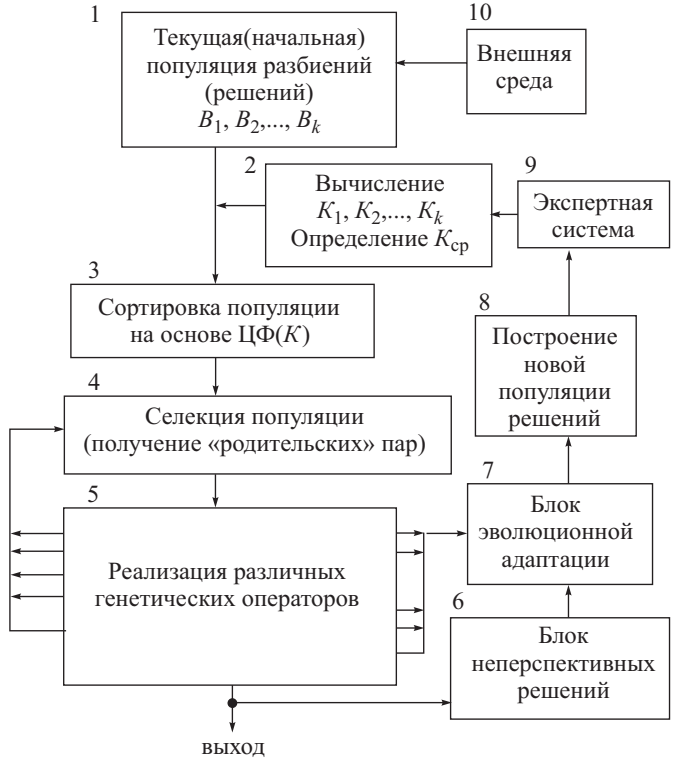


Рис. 5.1. Модифицированная схема генетического поиска для задачи разбиения

При наличии большого количества элементов предпочтительными будут конструктивные, итерационные, случайные и поисковые алгоритмы для получения текущей популяции. В качестве целевой функции выберем K (выражение (5.2)). В некоторых исследованиях, например, предлагается при решении задачи разбиения графа на части не минимизировать K (число связей между частями разбиения), а максимизировать количество связей (критерий A) внутри частей разбиения. Известно, что критерии K и A являются взаимообратными, т. е. минимизация K максимизирует A и наоборот. Для каждого элемента текущей популяции вычислим K и определим $K_{\text{ср}}$ — среднее значение целевой функции на данной популяции (рис. 5.1). Для каждого элемента текущей популяции в блоке 2 вычисляется K и определяется $K_{\text{ср}}$ — среднее значение целевой функции на данной популяции. Блок 3 осуществляет сортировку популяции на основе целевой функции.

Здесь могут быть применены все существующие методы сортировки. Сначала расставляются элементы с наименьшим значением K и так далее по возрастанию. Блок 4 выполняет селекцию популяции для получения родительских пар. Селекция выполняется одним из методов, описанных выше. Блок 5 осуществляет реализацию генетических операторов и их модификаций. Блок 6 собирает и анализирует неперспективные решения задачи разбиения. Блок 7 реализует стратегии адаптации и на основе обратных связей выбирает модель эволюции, а также порядок использования и применения различных алгоритмов генетической оптимизации. В 8 блоке осуществляется построение новой популяции решений. Девятый блок позволяет управлять процессом поиска с помощью информирующих обратных связей. Далее для каждой хромосомы из новой популяции вычисляется K и выживают те элементы из старой и новой популяции, у которых $K \leqslant K_{\text{ср}}$. При этом в стандартном случае количество элементов в новой популяции не должно превышать число элементов в старой популяции.

Предложенная схема генетического поиска позволяет варьировать размер популяции от генерации к генерации, что позволяет частично предотвращать преждевременную сходимость алгоритма в задачах разбиения. Можно предложить большое число аналогичных схем генетического поиска для решения задач разбиения. Их эффективность проверяется экспериментальным путем.

Заметим, что такая стратегия поиска позволяет быстрее находить локально-оптимальные результаты. Это связано с параллельной обработкой множества альтернативных решений, причем в такой схеме возможно концентрировать поиск на получение более перспективных решений. Отметим, что периодически в каждой итерации генетического алгоритма можно проводить различные изменения в перспективных, неперспективных и других решениях.

3.2. Рассмотрим последовательный генетический алгоритм разбиения. Пусть задан граф $G = (X, U)$. Первоначально упорядочим все вер-

шины графа по возрастанию локальных степеней вершин. Это соответствует списку вершин, а также может соответствовать тривиальному разбиению, когда количество групп разбиения равно упорядоченным вершинам графа. Далее начинаем составлять пары вершин и для каждой пары вычислять оценку связности:

$$\delta_{i,j} = e_{i,j} - (e_{i,t} + e_{j,t}), \tag{5.3}$$

где $e_{i,j}$ — число ребер между вершинами x_i и x_j, образовавшими пару; $e_{i,t}$ и $e_{j,t}$ — число ребер, соединяющих выбранную пару со всеми остальными вершинами графа, причем $t = 1, 2, \ldots, n-2$.

Составим второй список, где будут оставлены такие пары вершин, у которых $\delta_{i,j} \geqslant 0$. Отметим, что если пары $\delta_{i,j} \geqslant 0$ не находятся, то возможно два случая. В первом строятся пары с наименее отрицательным $\delta_{i,j}$. Во втором случае не создаются пары из двух вершин, и сразу происходит переход к образованию групп из трех вершин. Далее процесс повторяется аналогично, пока не будет выполнено разбиение графа на заданное или произвольное число частей. Это основная идея стандартной процедуры разбиения. При небольшом количестве вершин она дает удовлетворительные результаты, так как практически здесь осуществляется полный перебор вариантов решений. С увеличением числа вершин использование напрямую такого подхода становится нецелесообразно. Для сокращения числа просматриваемых пар применим простые и модифицированные генетические операторы. Отметим, что вместо выражения (5.3) можно использовать выражение

$$\tau_i + \tau_j \geqslant \tau_t, \tag{5.4}$$

где $\tau_i + \tau_j$ — внутренние ребра разбиения, τ_t $(t = 1, 2, \ldots, n-2)$ — внешние ребра разбиения.

3.3. Рассмотрим пример. Пусть задан граф (рис. 5.2). Определим локальные степени вершин и упорядочим их в виде списка по возрастанию (табл. 5.1). Построим теперь пары вершин для определения разбиения графа по две вершины в каждой части.

Таблица 5.1

x_i	x_2	x_3	x_4	x_5	x_6	x_1
ρ_{xi}	5	5	5	5	4	4

На этом уровне выделим два блока разбиения: это вершины (x_1, x_3), (x_2, x_6). Для удобства записи вместо $(x_1, x_2, \ldots)$ будем иногда записывать их соответствующие номера $(1, 2, \ldots)$. Если стоит задача разбить граф на части по две вершины в каждой с минимизацией K, то в качестве третьего блока разбиения можно взять блок с наименьшим отрицательным значением выражения (5.3). Такое разбиение показано на рис. 5.3.

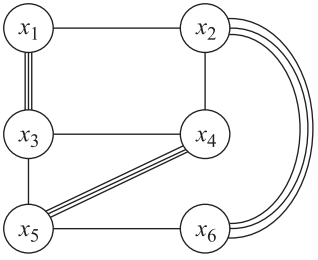


Рис. 5.2. Граф G

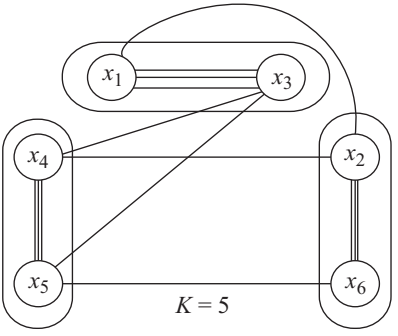


Рис. 5.3. Разбиение графа G на три части

В табл. 5.2 приведены все пары, которые можно получить, объединяя вершины данного графа. Очевидно, что при большом числе вершин эта процедура нецелесообразна. Применим одну из модифицированных схем генетического поиска, описанных в предыдущих разделах.

Таблица 5.2

$(x_i x_j)$	$(1,2)$	$(1,3)$	$(1,4)$	$(1,5)$	$(1,6)$	$(2,3)$	$(2,4)$	$(2,5)$	$(2,6)$
$\tau_{i+}\tau_j \geqslant \tau_t$	$1 \geqslant 7$	$3 \geqslant 3$	$0 \geqslant 9$	$0 \geqslant 9$	$0 \geqslant 8$	$0 \geqslant 10$	$1 \geqslant 8$	$0 \geqslant 10$	$3 \geqslant 3$
$(x_i x_j)$	$(3,4)$	$(3,5)$	$(3,6)$	$(4,5)$	$(4,6)$	$(5,6)$			
$\tau_{i+}\tau_j \geqslant \tau_t$	$1 \geqslant 8$	$1 \geqslant 8$	$0 \geqslant 9$	$3 \geqslant 4$	$0 \geqslant 9$	$1 \geqslant 7$			

Построим случайным образом некоторую популяцию решений $P = \{P_1, P_2, P_3, P_4, P_5, P_6\}$, $P_1 : (1,2)$. Теперь проверим выполнение условия (5.4) для всех элементов популяции P. Для первого элемента $1 \geqslant 7$ (условие не выполняется). Продолжая далее, получим

$$P_1 : (1,2) - 1 \geqslant 7, \qquad P_3 : (3,4) - 1 \geqslant 8, \qquad P_5 : (5,6) - 1 \geqslant 7,$$
$$P_2 : (2,3) - 0 \geqslant 10, \qquad P_4 : (4,5) - 3 \geqslant 4, \qquad P_6 : (3,6) - 0 \geqslant 9.$$

В качестве целевой функции взято выражение (5.4). Определим для каждой хромосомы в популяции значение целевой функции. Произведем сортировку анализируемой популяции и получим

$$P_4, \quad P_5, \quad P_1, \quad P_3, \quad P_6, \quad P_2.$$

Произведем селекцию и образуем родительские пары:

$$P_4 : (4,5) \qquad P_1 : (1,2) \qquad P_6 : (3,6)$$
$$P_5 : (5,6) \qquad P_3 : (3,4) \qquad P_2 : (2,3)$$

Применим стандартный оператор кроссинговера и получим новые хромосомы:

$P_4 : (4,5)$	$P_1 : (1,2)$	$P_6 : (3,6)$
$P_5 : (5,6)$	$P_3 : (3,4)$	$P_2 : (2,3)$
$P'_4 : (4,6) - 0 \geqslant 9$	$P'_1 : (1,4) - 0 \geqslant 9$	$P'_6 : (3,3)$
$P'_5 : (5,5)$	$P'_3 : (3,2) - 0 \geqslant 10$	$P'_2 : (2,6) - 3 \geqslant 3$ (да).

В результате однократного применения оператора кроссинговера мы получили лучшее решение $P'_2 : (2,6)$. Тенденции преимущества генетического алгоритма видны даже на данном простом примере.

Использование стандартного оператора кроссинговера может приводить к несуществующим решениям (разбиения P'_5, P'_6). Поэтому можно предложить несколько модификаций оператора кроссинговера. В частности, при скрещивании пар потомки могут получаться путем объединения первых и вторых элементов родителей с анализом полученных решений. При этом повторяющиеся элементы устраняются, а отсутствующие добавляются. Тогда получим

$P''_4 : (4,5)$	$P''_1 : (1,3)$	$P''_6 : (3,2)$
$P''_5 : (5,6)$	$P''_3 : (2,4)$	$P''_2 : (6,3)$

Здесь, кроме потомков, повторяющих родителей (P''_4, P''_5, P''_2), находится элемент P''_1 с одним из лучших значений целевой функции ($3 \geqslant 3$). Продолжая аналогично, построим группы разбиения по три вершины.

На рис. 5.4 показано разбиение графа G (рис. 5.2) на две части по три вершины в каждой. При этом целевая функция $K = 5$. Отметим, что генетическая процедура поиска может быть применена как единственная, так и множественная оптимизационная процедура. Кроме того, генетические операторы и операторы поиска могут быть вставлены в структурную схему генетического поиска как вспомогательные блоки для повышения качества последовательных алгоритмов.

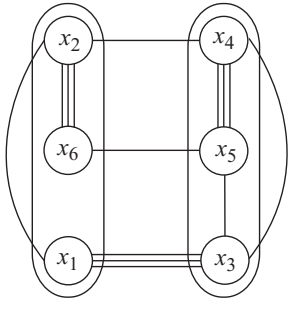


Рис. 5.4. Разбиение G на две части

В первом случае сложность алгоритма разбиения графа будет лежать в пределах $O(n)$–$O(n^3)$, причем крайний случай с $O(n^3)$ будет иметь место при популяции больше ста и выше тысячи генераций алгоритма.

3.4. Фракталами, как отмечалось выше, обычно называют множества, обладающие масштабной инвариантностью, т. е. в любом масштабе они выглядят практически одинаково. Для решения большинства нелинейных оптимизационных задач могут использоваться идеи фрактальных множеств.

Рассмотрим алгоритм разбиения графов на части на основе модифицированной агрегации фракталов. Процесс создания кластеров основан на идеях построения минимальных и квазиминимальных массивов в графе или гиперграфе. ***Кластер*** — это часть графа $\Phi \subseteq G$, причем $\Phi = (X_1, U_1)$, $X_1 \subseteq X$, $U_1 \subseteq U$. Вершины кластера соединены внутренними ребрами с остальными вершинами $X \backslash X_1$ графа G. Внешние ребра не входят в подмножество ребер кластеров. Мощность подмножества внешних ребер кластера обозначим f. Кластер Φ_i называется минимальным, если для любого другого кластера Φ_j, $\Phi_j \subset \Phi_i$ выполняется условие $f_i \leqslant f_j$. Другими словами, удаление произвольных вершин из Φ_i $(\Phi_i \backslash X_m)$ приводит к новому кластеру с большим числом внешних ребер. По определению будем считать:

$$(\forall \Phi_i \subset G)(\Phi_i \neq \varnothing), (\forall x_i \in X) \quad (x_i \text{ — минимальный кластер}).$$

Это означает, что минимальный кластер не может быть пустым. Кроме того, в тривиальном частном случае каждая вершина графа образует минимальный кластер. Кластер будем называть квазиминимальным, если выполняется условие

$$f_i + \varepsilon \leqslant f_j,$$

где ε — коэффициент, определяющий, на какую величину можно увеличить число внешних ребер в минимальном кластере Φ_i. Он определяется в ходе анализа математических моделей графа на основе решения ЛПР или ЭС.

На рис. 5.5, *а* показан минимальный кластер на три вершины с $f_1 = 4$. Соответственно на рис. 5.5, *б* оказан простой кластер с $f_2 = 9$. На рис. 5.6, *а* показан минимальный кластер на две вершины с $f_3 = 3$. При присоединении к этому минимальному кластеру вершины x_8 получим квазиминимальный кластер на три вершины с $f_4 = 5$ (рис. 5.6, *б*). В этом случае величина $\varepsilon = 1$.

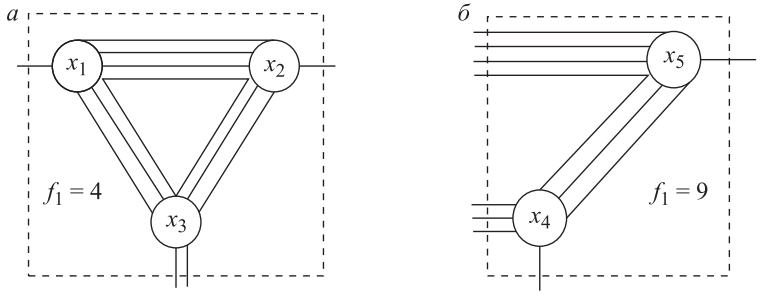


Рис. 5.5. *а* — минимальный кластер на три вершины, *б* — кластер на две вершины

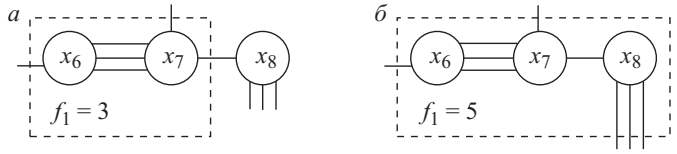


Рис. 5.6. *а* — минимальный кластер на две вершины, *б* — квазиминимальный кластер на три вершины

4. Построение минимального и квазиминимального кластеров связано с выделением почти всех подмножеств множества X, а оно составляет 2^n, если $|X| = n$. В этой связи продолжают разрабатываться различные эвристики выделения набора простых кластеров, минимального и квазиминимального. Эти эвристики основаны на теоремах о минимальных массивах в графах и принципах построения квазиминимального кластера.

Теорема 5.1. *Если* Φ_i *и* Φ_j *являются минимальными кластерами и* $\Phi_i \not\subset \Phi_j \wedge \Phi_j \not\subset \Phi_i$, *то* $\Phi_i \cap \Phi_j = \varnothing$.

Теорема 5.2. *Пусть* Φ_i, Φ_j — *минимальные кластеры. Если*

$$\Phi_s = \Phi_i \cup \Phi_j \wedge f_s < f_i \wedge f_s < f_j,$$

то Φ_s — *минимальные кластеры. Если*

$$\Phi_s = \Phi_i \cup \Phi_j \wedge f_s + \varepsilon \leqslant f_i \wedge f_s \leqslant f_j,$$

то Φ_s — *квазиминимальный кластер.*

Теорема 5.3. *Пусть* $\Phi_1, \Phi_2, \ldots, \Phi_L$ — *минимальные кластеры, причем любое объединение части из них не является минимальным кластером. Если*

$$\Phi_i = \Phi_1 \cup \Phi_2 \cup \ldots \cup \Phi_L \wedge f_i < f_1 \wedge f_i < f_2, \ldots, \wedge f_i < f_L,$$

то Φ_i — *минимальные кластеры. Если*

$$\Phi_i = \Phi_1 \cup \Phi_2 \cup \ldots \cup \Phi_L \wedge f_i + \varepsilon \leqslant f_1 \wedge f_i + \varepsilon \leqslant f_2, \ldots, \wedge \wedge f_i + \varepsilon \leqslant f_L,$$

то Φ_i — *минимальные кластеры.*

4.1. Из теоремы 5.1 следует, что в общем случае при полном переборе процедура построения минимального кластера сходящаяся и решение единственно при разбиении графа на части, когда не задано общее число вершин в частях разбиения. При разбиении графа на заданное число вершин количество решений в общем случае — это мощность некоторого подмножества альтернатив разбиений графа. На основе теорем 5.2 и 5.3 возможно агрегировать (объединять) кластеры и строить минимальные и квазиминимальные кластеры. Опишем модифицированный эвристический алгоритм построения минимальных и квазиминимальных кластеров в графе.

1. Упорядочить все локальные степени вершин графа как тривиальные минимальные и квазиминимальные кластеры.
2. Начиная с вершин с большей локальной степенью, попарно проанализировать все вершины графа G. Если определяется пара вершин $(x_i,\, x_j)$, для которой

$$f = \rho(x_i) + \rho(x_j) - 2r_{i,j},$$

то $\Phi = (X_1, U_1)$, $X_1 = \{x_i, x_j\}$, и он является минимальным кластером. Здесь $\rho(x_i)$, $\rho(x_j)$ — локальные степени вершин x_i, x_j; $r_{i,j}$ — число ребер, соединяющих вершины x_i, x_j между собой; f — число ребер, соединяющих минимальный кластер с остальными вершинами графа G, причем $f < \rho(x_i)$, $f < \rho(x_j)$. Если определяется пара вершин (x_i,x_j), для которой

$$f = \rho(x_i) + \rho(x_j) - 2r_{i,j} + \varepsilon,$$

причем $f + \varepsilon \leqslant \rho(x_i)$, $f + \varepsilon \leqslant \rho(x_j)$, то $\Phi = (X_1, U_1)$, $X_1 = \{x_i, x_j\}$, и Φ является квазиминимальным кластером. Минимальные или квазиминимальные кластеры Φ заносим в специальный список, а вершины x_i, x_j исключаем из рассмотрения. Перейти к 2. Если новых минимальных или квазиминимальных кластеров не образовалось, то переход к 3.
3. Выполнить факторизацию. Вершины x_i, x_j, входящие в минимальные или квазиминимальные кластеры, заменить одной вершиной $x_{i,j}$. При этом ребра, соединяющие x_i и x_j, удалить, а ребра из вершин x_i, x_j к другим вершинам приписать общей вершине $x_{i,j}$. Получается граф G'.
4. В графе G' проанализировать все вершины. Если определится тройка вершин x_A, x_B, x_C, для которой справедливо

$$f = \rho(x_\mathrm{A}) + \rho(x_\mathrm{B}) + \rho(x_\mathrm{C}) - 2r_{\mathrm{A,B}} - 2r_{\mathrm{B,C}} - 2r_{\mathrm{A,C}},$$

причем $f < \rho(x_\mathrm{A})$, $f < \rho(x_\mathrm{B})$, $f < \rho(x_\mathrm{C})$, то вершина x_A, x_B, x_C образует минимальный кластер согласно теореме 5.3. Если определится тройка вершин x_A, $x_\mathrm{B}, x_\mathrm{C}$, для которой справедливо

$$f = \rho(x_\mathrm{A}) + \rho(x_\mathrm{B}) + \rho(x_\mathrm{C}) - 2r_{\mathrm{A,B}} - 2r_{\mathrm{B,C}} - 2r_{\mathrm{A,C}} + \varepsilon,$$

причем $f + \varepsilon \leqslant \rho(x_\mathrm{A})$, $f + \varepsilon \leqslant \rho(x_\mathrm{B})$, $f + \varepsilon \leqslant \rho(x_\mathrm{C})$, то вершины x_A, x_B, x_C образуют квазиминимальный кластер согласно теореме 5.3. Перейти к 2. Минимальный или квазиминимальный кластер занести в список. Если минимальных или квазиминимальных кластеров больше нет, то перейти к 4.
5. Увеличить параметр мощности минимального или квазиминимального кластера на единицу и перейти к 3. Если мощность минимального или квазиминимального кластера равна заданной величине или равна числу вершин графа, то перейти к 5.
6. Конец работы алгоритма.

Здесь ε — наименьшее относительное отклонение от минимального кластера ($\varepsilon = 0, 1, 2, \ldots$). После построения набора минимальных или квазиминимальных кластеров, если в графе G остались свободные вершины, необходимо провести процедуру агрегации. Она заключается в пробном помещении вершин в минимальный или квазиминимальный кластер с анализом значений целевой функции. Вершина помещается в минимальный кластер, если нарушение минимального кластера происходит на величину, не превышающую ε. Если после второй процедуры остались свободные вершины, реализуются процедуры, основанные на моделировании эволюций Дарвина, де Фризе, Ламарка и Поппера, описанные выше. Предпочтение отдается модифицированным операторам мутации, инверсии, сегрегации, удаления и вставки.

4.2. Рассмотрим пример, показанный на рис. 5.7. Приведен граф $G = (X, U)$, $|X| = 14$, $|U| = 37$, который должен быть разбит на части с наименьшим значением K (5.2).

Отметим, что число частей разбиения заранее не задано и определяется в процессе реализации алгоритма после каждого шага построения минимальных и квазиминимальных кластеров.

Матрица смежности R графа G имеет вид:

$$R = \begin{array}{c|cccccccccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} \\ \hline x_1 & 0 & 0 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ x_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ x_3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 2 & 1 & 0 & 0 \\ x_4 & 1 & 0 & 0 & 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_5 & 1 & 0 & 0 & 2 & 0 & 1 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ x_6 & 2 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_7 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 1 \\ x_9 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 \\ x_{10} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 2 & 0 \\ x_{11} & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ x_{12} & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 0 \\ x_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 2 & 0 & 0 & 0 & 0 \\ x_{14} & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}.$$

По матрице определим локальную степень каждой вершины и построим упорядоченный по возрастанию список степеней этих вершин. Сначала определяется минимальный кластер $\Phi_1 = \{x_2, x_{14}\}$, при этом $K_1 = 2$, $|U_1| = 4$, $|U_1| - K_1 = 2 > 0$. Производится факторизация. Строки и столбцы x_2, x_{14} матрицы R представляются объединенной строкой и столбцом (x_2, x_{14}), т. е. вершины x_2, x_{14} с соединяющими их ребрами заменяются вершиной Φ_1, и продолжается поиск минимального или квазиминимального кластеров, состоящих из двух вершин. Таких минимальных или квазиминимальных кластеров больше нет. Переходим к поиску минимального или квазиминимального кластера, состоящих из трех вершин. Согласно алгоритму получим $\Phi_2 = \{x_8, x_{10}, x_{13}\}$, при этом $K_2 = 1$, $|U_2| = 6$, $|U_2| - K_2 = 5 > 0$. При определении K_2 не

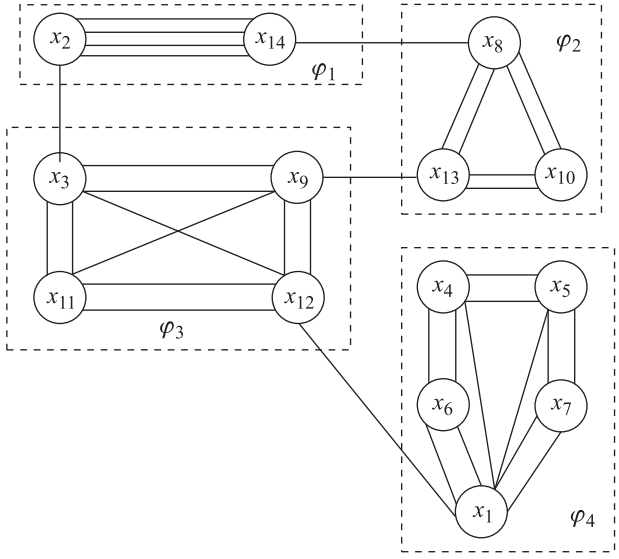


Рис. 5.7. Граф G

учитываются связи Φ_1 с Φ_2, чтобы не определять одни и те же внешние ребра дважды. Продолжая реализацию алгоритма, аналогично получим: $\Phi_3 = \{x_3, x_9, x_{11}, x_{12}\}$, $K_3 = 1$, $|U_3| = 10$, $|U_3| - K_3 = 9 > 0$ и $\Phi_4 = \{x_4, x_5, x_6, x_7, x_1\}$, $K_4 = 0$, $|U_4| = 16$, $|U_4| - K_4 = 16 > 0$. Общее число внешних связей между частями разбиения графа G (рис. 5.7) равно $K = K_1 + K_2 + K_3 + K_4 = 2 + 1 + 1 + 0 = 4$.

4.3. Рассмотрим другой пример, показанный на рис. 5.8.
Матрица смежности графа G имеет вид

$$R = \begin{array}{c|cccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \hline 1 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 0 & 1 & 0 & 0 & 5 & 0 & 0 & 0 & 2 & 0 & 0 \\ 6 & 0 & 0 & 0 & 1 & 5 & 0 & 5 & 1 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 1 & 0 & 5 & 0 & 0 & 1 & 0 & 1 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 & 0 & 3 \\ 12 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \end{array}.$$

По матрице определим локальную степень каждой вершины и построим упорядоченный по возрастанию список степеней этих вершин. В результате выполнения алгоритма определим минимальный кластер:

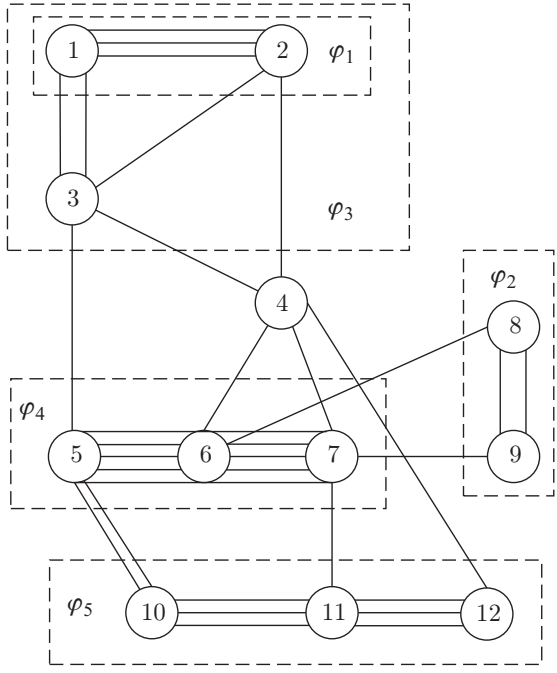


Рис. 5.8. Граф G

$\Phi_1 = \{x_1, x_2\}$, $\Phi_2 = \{x_8, x_9\}$. Представив Φ_1 и Φ_2 в виде вершин, получим новый граф, матрица смежности которого примет вид

$$R_1 = \begin{array}{c|cccccccccc} & (1,2) & 3 & 4 & 5 & 6 & 7 & (8,9) & 10 & 11 & 12 \\ (1,2) & 0 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 5 & 0 & 1 & 0 & 0 & 5 & 0 & 0 & 2 & 0 & 0 \\ 6 & 0 & 0 & 1 & 5 & 0 & 4 & 1 & 0 & 0 & 0 \\ 7 & 0 & 0 & 1 & 0 & 4 & 0 & 1 & 0 & 1 & 0 \\ (8,9) & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 3 & 0 \\ 11 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 & 3 \\ 12 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 0 \end{array}.$$

Так как минимальных кластеров мощности 2 больше нет, то построим согласно алгоритму минимальный кластер мощности 3. Получим $\Phi_3 = \{\Phi_1, x_3\}$, $\Phi_4 = \{x_5, x_6, x_7\}$, $\Phi_5 = \{x_{10}, x_{11}, x_{12}\}$. Работа алгоритма закончена, больше минимальных кластеров нет. Вершину x_4, которая, вообще говоря, есть тривиальный минимальный кластер, можно случайным образом добавлять во все остальные минимальные кластеры с вычислением значения целевой функции. В результате получим два варианта разбиения:

1-й вариант:

$$\{x_1, x_2, x_3, x_4\},\ \{x_5, x_6, x_7\},\ \{x_8, x_9\},\ \{x_{10}, x_{11}, x_{12}\};$$

при этом $K = 9$.

2-й вариант:

$$\{x_1, x_2, x_3\},\ \{x_4, x_5, x_6, x_7\},\ \{x_8, x_9\},\ \{x_{10}, x_{11}, x_{12}\};$$

при этом $K = 9$.

Например, если задано разбиение графа на три части, вариант 1 после процедуры эволюции запишется следующим образом.

3-й вариант:

$$\{x_1, x_2, x_3, x_4\},\ \{x_5, x_6, x_7, x_8, x_9\},\ \{x_{10}, x_{11}, x_{12}\};$$

в этом случае $K = 7$.

Рассмотрим основную идею итерационного разбиения гиперграфа $h = (x, E)$ на части с минимизацией K. Для каждого ребра $e_i \in E$ можно определить разрезающую функцию как число вершин дерева, моделирующих анализируемую цепь коммутационной схемы электронно-вычислительной аппаратуры, попадающих в различные блоки. Если хотя бы одна вершина ребра гиперграфа лежит не в той части, что остальные вершины этого ребра, то обязательно будет существовать, по крайней мере, одно разрезающее ребро. Просуммировав все разрезающие ребра, получим общее число K (связей) между блоками, которое необходимо минимизировать.

Предлагаемый алгоритм состоит из трех этапов. На первом этапе строится набор популяций, который может быть получен случайными, последовательными или итерационными методами. На втором этапе для каждой хромосомы в популяции определяется целевая функция. В качестве целевой функции выбирается K. Затем полученная популяция сортируется и упорядочивается согласно целевой функции. Далее выполняется разбиение популяции на две подпопуляции: перспективных и неперспективных хромосом. Для первой подпопуляции используется точный поиск, т. е. применяются различные архитектуры генетического алгоритма, описанные выше. Для второй подпопуляции используется быстрый поиск — простой генетический алгоритм. Затем реализуется оператор миграции хромосом между подпопуляциями. На третьем этапе, используя описанные схемы селекции, выбираются родительские пары или отдельные хромосомы для выполнения генетических операторов.

Перед выполнением генетических операторов в каждой хромосоме (текущий вариант разбиения) определяется стоимость существующих строительных блоков. Работа генетических операторов начинается с модифицированного оператора кроссинговера. При этом точка или точки оператора кроссинговера определяются не случайно, а направленно, выделяя лучшие строительные блоки в каждой хромосоме.

Например, на рис. 5.9 показана схема модифицированного оператора кроссинговера. Здесь P_1, P_2 — хромосомы из популяции P. Строительные блоки, т. е. части разбиения, следующие: $\{a, b, c\}$, $\{d, e\}$, $\{f, g\}$; $\{d, e, c\}$, $\{a, f\}$, $\{b, g\}$. Определяя целевую функцию, найдем, что K_2 и K_5 — наименьшие. Тогда точки оператора кроссинговера в P_1 направленным образом выделяют блок $\{d, e\}$, а в P_2 — блок $\{a, f\}$. Затем производится копирование лучших строительных блоков из P_1 и P_2 в P_1'. Свободные места в P_1' заполняются оставшимися элементами из P_1 и P_2.

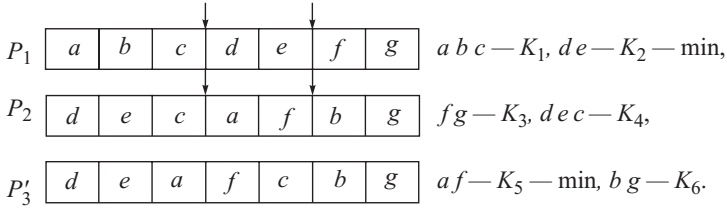


Рис. 5.9. Схема модифицированного оператора кроссинговера

4.4. Возможны случаи, когда выбранные строительные блоки могут иметь частичное или полное совпадение элементов. При полном совпадении элементов производится копирование только одного из строительных блоков. При частичном совпадении — строительный блок из первого родителя копируется полностью, а из второго родителя копируются несовпадающие элементы. Оператор мутации может выполняться случайно и направленно. На рис. 5.10 показано возможное применение оператора мутации, причем элементы могут мутировать только между строительными блоками. При направленной мутации в каждом строительном блоке выбирается наихудший элемент.

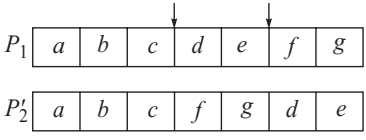


Рис. 5.10. Пример оператора мутации

4.5. Для выхода из локальных оптимумов предлагаются две процедуры. В основе первой лежат модифицированные операторы сегрегации, транслокации, удаления и вставки. На рис. 5.11 показан пример данного оператора. Заметим, что строительный блок из хромосом копируется полностью, если они не пересекаются с уже выделенными блоками. В противном случае копируются только непересекающиеся элементы из строительного блока.

Важную роль в генетических операторах, как отмечалось выше, играет вероятность оператора кроссинговера Pr(ОК) и вероятность мутации Pr(ОМ). Эксперименты показали, что в задачах разбиения Pr(ОК), лежащая в пределах $0{,}5 \div 0{,}98$, и Pr(ОМ) в пределах $0{,}01 \div 0{,}05$, позволяют получать большое число локальных оптимумов за $1000 \div 10000$ генераций.

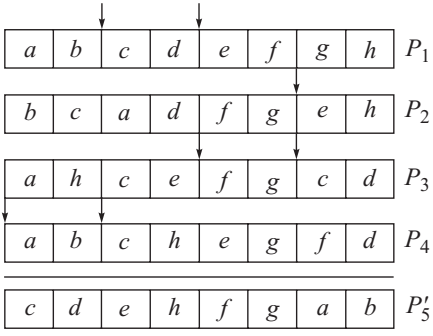


Рис. 5.11. Пример оператора сегрегации

В основе второй процедуры лежат идеи миграции хромосом внутри одной популяции и между популяциями. В зависимости от наличия вычислительных ресурсов можно первоначально строить не одну, а некоторый набор популяций. Далее выбирать «лучшего» представителя из каждой популяции и на этом множестве осуществлять поиск.

4.6. Определим асимптотическую трудоемкость данного алгоритма. Пусть N_p — размер популяции, $N_{\text{п}}$ — число полученных потомков, N_G — число генераций генетического алгоритма. Тогда общую трудоемкость алгоритма можно ориентировочно определить

$$T \approx [\alpha N_p t_p + \beta N_{\text{п}} t_{\text{п}} + \gamma (N_p + N_{\text{п}}) t_{\text{с}}] N_G,$$

где t_p — трудоемкость построения одной хромосомы в популяции; $t_{\text{п}}$ — трудоемкость генерации одного потомка; $t_{\text{с}}$ — совокупная трудоемкость селекции и отбора; α, β, γ — коэффициенты, определяющие параметры генетического поиска. Заметим, что трудоемкость построения одной хромосомы может колебаться от $O(n)$ до $O(n!)$, а трудоемкость генерации одного потомка зависит от сложности применяемого генетического оператора и ориентировочно изменяется от $O(n)$ до $O(n \log n)$. Трудоемкость селекции и отбора может меняться от $O(n)$ до $O(n^2)$. Временная сложность описанного алгоритма ориентировочно составляет $O(n^2)$ для одной генерации.

Следует заметить, что с увеличением числа генераций в генетическом алгоритме время решения оптимизационной задачи разбиения повышается, но это повышение незначительное и может быть компенсировано получением множества локально-оптимальных решений. Генетические алгоритмы требуют бóльших затрат времени, чем алгоритмы парных перестановок и случайного разбиения графов. Но они позволяют получать набор локально-оптимальных решений.

4.7. Допустимое решение экстремальной задачи разбиения графа на части является n-мерным вектором. В том случае, когда эта задача принадлежит классу задач переборного типа, имеется конечное множество допустимых решений, в которых каждая компонента вектора может быть закодирована с помощью целого неотрицательного числа: $\beta_i \in [0, V_i]$, $i = \overline{1, n}$, где $(V_i + 1)$ — число возможных дискретных значений i — управляемой переменной в области поиска D.

Это позволяет поставить во взаимнооднозначное соответствие каждому вектору $P_i \in D$ вектор $\boldsymbol{\beta}$ с целочисленными компонентами. Согласно Д. Батищеву, символьная модель экстремальной задачи разбиения графа на части может быть представлена в виде множества бинарных строк, которые описывают конечное множество допустимых решений, принадлежащих области поиска D. Необходимо отметить, что выбор символьной модели исходной экстремальной задачи разбиения графа на части во многом определяет эффективность и качество применяемых генетических алгоритмов.

Представим, например, разбиение графа $G = (X, U)$ на две части в виде бинарной строки $E(X_1, X_2)$, состоящей из n бит, расположенных в порядке возрастания их номеров. Каждому номеру бита поставим во взаимнооднозначное соответствие номер вершины графа (1-й бит соответствует вершине x_1, 2-й бит — вершине $x_2, \ldots$, n-й бит — вершине x_n). Потребуем, чтобы бинарное значение α_l 1-го бита указывало, какому подмножеству вершин (X_1 или X_2) принадлежит вершина x_l:

$$\alpha_l = \begin{cases} 1, & \text{если } l\text{-я вершина } x_l \in X \text{ входит в состав} \\ & \text{подмножества вершин } X_1; \\ 0, & \text{если } l\text{-я вершина } x_l \in X \text{ входит в состав} \\ & \text{подмножества вершин } X_2. \end{cases}$$

Число битов, содержащих «1» в бинарной строке $E(X_1, X_2)$, должно равняться мощности подмножества вершин подграфа $G_1 = (X_1, U_1)$, равной порядку этого подграфа n_1. Для графа G (рис. 5.4) получим следующую кодировку разбиения в виде бинарных строк:

$E(X_1, X_2)$:	1	1	0	0	0	1	
	1	2	3	4	5	6	— номер бита
	x_1	x_2	x_3	x_4	x_5	x_6	— номер вершины

В задаче оптимального разбиения графа на части в качестве хромосомы выступает конкретное разбиение, удовлетворяющее заданным условиям, что позволяет интерпретировать решение задачи разбиения как эволюционный процесс, связанный с перераспределением вершин $x_i \in X$ графа G по частям разбиения.

4.8. Для описания популяций Д. Батищев ввел два типа вариабельных признаков, отражающих качественные и количественные различия между хромосомами. Качественные признаки — признаки, которые позволяют однозначно разделять совокупность хромосом на четко различимые группы. Количественные признаки — признаки, проявляющие непрерывную изменчивость.

Кратко опишем генетический алгоритм дихотомического разбиения графа G=(X, U) на два подграфа: $G_1 = (X_1, U_1)$ и $G_2 = (X_2, U_2)$, $X_1 \cup X_2 = X$, $U_1 \cup U_2 \cup U_3 = U$, где U_3 — подмножество ребер, соединяющих два подграфа, причем для любого ребра $U_i \in U_3$, $U_i = (x_i, x_j)$,

$x_i \in X_1$ и $x_j \in X_2$ или $x_i \in X_2$ и $x_j \in X_1$. Целевая функция или критерий развития — это число ребер, соединяющих подграфы ($K = |U_3|$). Тогда оптимальным является решение (X_1^*, X_2^*) следующей экстремальной задачи однокритериального вектора:

$$Q^* = Q(X_1^*, X_2^*) = \min K, \quad (X_1, X_2 \subset X),$$

причем $K = |U \backslash (U_1 \cup U_2)|$, где U_1, U_2 — подмножества ребер, соединяющих между собой только вершины $X_1(X_2)$.

При взаимодействии хромосомы с внешней средой ее генотип порождает совокупность внешне наблюдаемых количественных признаков, включающих степень приспособленности $\mu(P_k)$ хромосомы P_k к внешней среде и ее фенотип. Приняв в качестве внешней среды критерий оптимальности K, видим, что степенью приспособленности $\mu(P_k)$ каждой хромосомы является численное значение функции K, определенное для допустимого решения $P_k \in D$. В общем случае степень приспособленности $\mu(P_k) \geqslant 0$ согласно Д. Батищеву можно задать с помощью следующего выражения:

$$\mu(P_k) = \begin{cases} Q^2(\mathbf{x}), & \text{при максимизации функции,} \\ \dfrac{1}{Q^2(\mathbf{x}) + 1}, & \text{при минимизации функции.} \end{cases}$$

Из данного выражения следует, что чем больше численное значение степени приспособленности $\mu(P_k)$, тем лучше хромосома приспособлена к внешней среде. Следовательно, цель эволюции хромосом заключается в повышении их степени приспособленности. Степень приспособленности $\mu(P_k)$ в задаче разбиения просто совпадает с критерием оптимальности K — общей суммой числа внешних ребер между частями разбиения.

4.9. Очевидно, что в популяции P^t может иметь место наличие нескольких различающихся форм того или иного вариабельного признака (так называемый полиморфизм), что позволяет проводить разделение популяции на ряд локальных подпопуляций $P_i^t \subset P^t$, $i = \overline{1, V}$, включающих в свой состав те хромосомы, которые имеют одинаковые или «достаточно близкие» формы тех или иных качественных и/или количественных признаков.

Так, в задаче оптимального разбиения графа на части для дифференциации хромосом по количественному признаку может быть выбрано, например, условие, что в локальную популяцию $P_1^t \subset P^t$ включаются только те хромосомы, у которых значение K не превосходит некоторой заданной величины. Тогда другую локальную популяцию $P_2^t \subset P^t$ составят все те хромосомы P_k, которые не попали в P_1^t.

4.10. При разбиении графов на части возникают задачи группировки элементов, обладающих одинаковыми свойствами. Для решения таких задач будем использовать идеи построения квазиминимальных

кластеров на основе агрегации (объединения) фракталов. Моделью исходных данных задачи, может, например, быть матрица следующего вида:

$$R = \begin{array}{c|c} & \begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{array} \\ \hline \begin{array}{c} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \end{array} & \begin{array}{cccccccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \end{array}.$$

Здесь столбцы матрицы R соответствуют заданным элементам графа, а строки — свойствам или выполненным функциям этих элементов. Например, единица на пересечении строки A_4 и столбца 4 означает, что элемент 4 обладает свойствами A_4. Необходимо сгруппировать элементы, обладающие наибольшим числом одинаковых свойств. Данная задача относится к классу задач разбиения и может быть решена одним из эвристических алгоритмов.

Приведем основную идею предлагаемого эвристического алгоритма. Предварительно выполняется анализ матричной модели. В результате анализа предлагается удалить из рассмотрения строки матрицы, содержащие все единицы, все нули, а также строки, содержащие около 1 % нулей или единиц. Такое удаление не будет влиять на создание требуемых групп, так как все или почти все элементы будут входить или не входить в формируемые группы. Тогда в рассматриваемом примере исключим строки A_5 и A_6. Далее произведем сортировку матрицы R. При этом на первое место поместим столбец с наибольшим числом единиц и так далее по убыванию.

В этом случае матрица R запишется так:

$$R_1 = \begin{array}{c|c} & \begin{array}{ccc|ccc|ccc|ccc} 12 & 4 & 7 & 8 & 9 & 11 & 1 & 5 & 6 & 2 & 3 & 10 \end{array} \\ \hline \begin{array}{c} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_6 \\ A_7 \end{array} & \begin{array}{ccc|ccc|ccc|ccc} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{array} \end{array}.$$

Затем производится разбиение матрицы R на две части: R_1 «перспективных» хромосом и R_2 «неперспективных» хромосом. Разбиение, включающее элементы $\{12, 4, 7, 8, 9, 11\}$, относится к «перспективным», а разбиение $\{1, 5, 6, 2, 3, 10\}$ — к «неперспективным». В свою очередь, для распараллеливания процесса части R_1 и R_2 можно снова разбить на две части: R_1', R_1'' и R_2', R_2''. Данный процесс можно продолжать до получения подмножеств разбиения, которые легко поддаются обработке генетическими операторами. Предполагается процедура поэлементного сложения столбцов матрицы R по следующим правилам: $1 + 1 = 1$; $1 + 0 = 0$; $0 + 0 = 0$. Например, выполняя указанные сложения для

«перспективных» столбцов матрицы R внутри частей R'_1 и R''_1 $(12+4;\ 12+7\ ;\ 4+7)$, получим

$$\begin{array}{r} 12: \ 1\ 1\ 1\ 1\ 0\ 1 \\ +\ 4: \ 0\ 1\ 0\ 1\ 1\ 1 \\ \hline (12,4): \ 0\ 1\ 0\ 1\ 0\ 1 \end{array} \quad \begin{array}{r} 12: \ 1\ 1\ 1\ 1\ 0\ 1 \\ +\ 7: \ 1\ 1\ 1\ 0\ 0\ 1 \\ \hline (12,7): \ 1\ 1\ 1\ 0\ 0\ 1 \end{array} \quad \begin{array}{r} 4: \ 0\ 1\ 0\ 1\ 1\ 1 \\ +\ 7: \ 1\ 1\ 1\ 0\ 0\ 1 \\ \hline (4,7): \ 0\ 1\ 0\ 0\ 0\ 1 \end{array}$$

В группе $(12,4)$ — 3 общих свойства, в группе $(12,7)$ — 4 общих свойства и в группе $(4,7)$ — 2 общих свойства. Продолжая далее, согласно процедурам агрегации фракталов, получим

$$\begin{array}{r} 8: \ 1\ 1\ 0\ 1\ 1\ 0 \\ +\ 9: \ 1\ 0\ 1\ 0\ 1\ 1 \\ \hline (8,9): \ 1\ 0\ 0\ 0\ 1\ 0 \end{array} \quad \begin{array}{r} 8: \ 1\ 1\ 0\ 1\ 1\ 0 \\ +\ 11: \ 1\ 1\ 1\ 0\ 0\ 1 \\ \hline (8,11): \ 1\ 1\ 0\ 0\ 0\ 0 \end{array} \quad \begin{array}{r} 9: \ 1\ 0\ 1\ 0\ 1\ 1 \\ +\ 11: \ 1\ 1\ 1\ 0\ 0\ 1 \\ \hline (9,11): \ 1\ 0\ 1\ 0\ 0\ 1 \end{array}$$

В группе $(8,9)$ — 2 общих свойства, в группе $(8,11)$ — 2 общих свойства и в группе $(9,11)$ — 3 общих свойства. Фиксируем группы, имеющие три и более общих свойств. Выполняя аналогичные преобразования внутри «перспективных» частей R'_2 и R''_2, получим следующие группы $(1,5)$ — 2 общих свойства; $(1,6)$ — 1; $(5,6)$ — 1; $(2,3)$ — 0; $(2,10)$ — 1; $(3,10)$ — 0. Производя сортировку полученных решений в группах, сформируем следующую популяцию:

$$P = \{(12,7), (12,4), (9,11), (4,7), (8,9), (8,11), (1,5), (2,10)\}.$$

Для реализации совместной модели эволюции сформируем следующие родительские пары и выполним оператор кроссинговера:

$$\begin{array}{r} P_1: \ 12 \ | \ 7 \\ P_3: \ 9 \ | \ 11 \\ \hline P'_1: \ 12 \ | \ 11 \\ P'_3: \ 9 \ | \ 7 \end{array} \quad \begin{array}{r} P_2: \ 12 \ | \ 4 \\ P_4: \ 4 \ | \ 7 \\ \hline P'_2: \ 12 \ | \ 7 \\ P'_4: \ 4 \ | \ 4 \end{array} \quad \begin{array}{r} P_5: \ 8 \ | \ 9 \\ P_7: \ 1 \ | \ 5 \\ \hline P'_5: \ 8 \ | \ 5 \\ P'_7: \ 1 \ | \ 9 \end{array} \quad \begin{array}{r} P_6: \ 8 \ | \ 11 \\ P_8: \ 2 \ | \ 10 \\ \hline P'_6: \ 8 \ | \ 10 \\ P'_8: \ 2 \ | \ 11 \end{array}$$

В группах после применения стандартного одноточечного оператора кроссинговера имеем $(12,11)$ — 4 общих свойства; $(9,7)$ — 3; P'_2 совпадает с P_1, поэтому не рассматривается, P'_4 — нелегальная группа; $(8,5)$ — 2; $(1,9)$ — 3; $(8,10)$ — 2; $(2,11)$ — 1. Отметим, что использование жадных и других модифицированных операторов кроссинговера позволит получать реальные решения.

Сформируем теперь новую популяцию:

$$P_1 = \{(12,7),\ (12,11),\ (9,11),\ (12,4),\ (9,7),\ (1,9),\ (8,5),\ (8,10)\}.$$

Здесь размер P_1 оставлен равным P. Как отмечалось выше, размер популяции можно варьировать в зависимости от качества получаемых решений. В принципе, можно выполнить некоторое множество генераций данной операции. Произведем теперь склеивание (объединение) групп по критерию максимума общих свойств. Тогда получим следу-

ющие группы: группа $(12, 7, 11)$ имеет четыре совпадающих элемента; $(12, 7, 9)$ — 3; $(12, 11, 9)$ — 3; $(7, 11, 9)$ — 3. Продолжая аналогично, получим группу $(12, 7, 11, 9, 1)$ с тремя совпадающими элементами.

4.11. Временная сложность данного алгоритма для одной генерации меняется от $O(n)$ до $O(n \log n)$. Данный алгоритм несложен и может выполняться параллельно. Ориентировочная трудоемкость алгоритма:

$$T \approx (N_p t_p + N_P t_{\text{ОК}} + N_p t_{\text{скл}}) N_G,$$

где $t_{\text{ОК}}$ — трудоемкость операторы кроссинговера, а $t_{\text{скл}}$ — трудоемкость операции склеивания.

Заметим, что с увеличением числа генераций в генетическом алгоритме время решения повышается, но это повышение незначительное и компенсируется получением множества локально-оптимальных решений. Генетические алгоритмы требуют больших затрат времени, чем алгоритмы парных перестановок и случайного разбиения графов. Но они позволяют получать набор локально-оптимальных решений. Генетические алгоритмы по скорости практически совпадают с итерационными и несколько хуже последовательных, причем они быстрее, чем метод ветвей и границ и метод отжига. В отличие от всех рассмотренных алгоритмов разбиения генетические позволяют получать набор квазиоптимальных результатов. Разбиение графов с применением описанных методов позволяет всегда получать локальные оптимумы, иметь возможность выхода из них и приближаться к получению оптимальных и квазиоптимальных решений, причем временная сложность алгоритма не уходит из области полиномиальной сложности ($\approx O(n \log n) \div O(n^3)$).

5.2. Решения задачи о коммивояжере

1. Задача о коммивояжере (в английской интерпретации TSP) является классической NP-полной задачей. Она заключается в нахождении кратчайшего гамильтонова цикла в графе. Без каких-либо изменений в постановке она используется для проектирования разводки коммуникаций, разработки архитектуры вычислительных сетей и др. Кроме того, задача о коммивояжере имеет теоретическую ценность, являясь асимптотической оценкой при исследовании различных эвристических алгоритмов, которые затем применимы для решения более сложных задач комбинаторной оптимизации, например, квадратичной задачи о назначениях, частным случаем которой является задача о коммивояжере.

Рассмотрим постановку задачи. В неформальной форме задача о коммивояжере трактуется следующим образом: коммивояжеру необходимо посетить W городов, не заезжая в один и тот же город дважды, и вернуться в исходный пункт по маршруту с минимальной стоимостью. Более строго, дан граф $G = (X, U)$, где $|X| = n$ — множество

вершин (городá), $|U| = m$ — множество ребер (возможные пути между городами), показанный на рис. 5.12. Дана матрица чисел $R(i, j)$, где $i, j \in 1, 2, \ldots, n$, представляющих собой стоимость переезда из вершины x_i в x_j.

Требуется найти перестановку φ из элементов множества X такую, что значение целевой функции равно

$$\texttt{Fitness}\,(\varphi)R(\varphi(1), \varphi(n)) + \sum_i \{R(\varphi(i), \varphi(i+1))\} \to \min.$$

Если граф не является полным, то дополнительными ограничениями на величину φ являются:

$$\langle \varphi(i), \varphi(i+1) \rangle \in U \quad \forall\, i \in 1, 2, \ldots, n-1 \text{ и } \langle \varphi(1), \varphi(n) \rangle \in U.$$

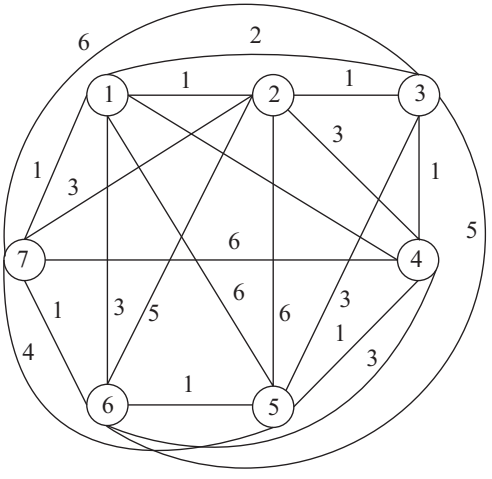


Рис. 5.12. Граф $G = (X, U)$

Данные ограничения учитываются проставлением метки бесконечности в матрицу R в ячейках, соответствующих отсутствующим ребрам в графе. Поэтому в дальнейшем при решении задачи о коммивояжере будем рассматривать полные графы с весами на ребрах.

Запишем матрицу смежности R графа $G = (X, U)$, где $|X| = 7$, $|U| = 21$ (рис. 5.12). Для пути $\varphi(1) = \langle 1, 2, 3, 4, 5, 6, 7 \rangle$ (хромосома P_1) имеем значение целевой функции:

$$\texttt{Fitness}\,(\varphi(1)) = 1 + 1 + 1 + 1 + 1 + 1 + 1 = 7,$$

а для пути $\varphi(2) = \langle 7, 2, 5, 4, 3, 6, 1 \rangle$ (хромосома P_2) имеем значение целевой функции:

$$\texttt{Fitness}\,(\varphi(2)) = 3 + 6 + 1 + 1 + 5 + 3 + 1 = 20.$$

Следовательно, маршрут $(\varphi(1))$ предпочтительнее, чем $(\varphi(2))$, с точки зрения нахождения минимального маршрута. Отметим, что зна-

чение целевой функции $\texttt{Fitness}(\varphi)$ не зависит в частном случае от выбора вершины — начала маршрута. Например, $\texttt{Fitness}(\varphi(3)) = \texttt{Fitness}(\varphi(2)) = 20$, где $\varphi(3) = \langle 2,3,4,5,6,7,1\rangle$ (хромосома P_3), являясь инвариантом для всех возможных циклических сдвигов. Следовательно, в графе существует большое число равнозначных маршрутов:

$$R = \left|\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 1 & 2 & 7 & 6 & 3 & 1 \\ 2 & 1 & 0 & 1 & 3 & 6 & 5 & 3 \\ 3 & 2 & 1 & 0 & 1 & 3 & 5 & 6 \\ 4 & 7 & 3 & 1 & 0 & 1 & 3 & 6 \\ 5 & 6 & 6 & 3 & 1 & 0 & 1 & 4 \\ 6 & 3 & 5 & 5 & 3 & 1 & 0 & 1 \\ 7 & 1 & 3 & 6 & 6 & 4 & 1 & 0 \end{array}\right|.$$

Фиксация вершины-старта может быть использована для сужения пространства поиска, которое в этом случае равно $(n-1)!$, поэтому в общем случае задача о коммивояжере чувствительна к выбору начальных условий. В этой связи для создания структуры алгоритма эффективно используются генетические алгоритмы, описанные в предыдущих разделах.

2. Время работы алгоритмов, основанных на полном переборе или на построении дерева решений (методы ветвей и границ, золотого сечения и др.), растет экспоненциально в зависимости от числа вершин графа. Эти алгоритмы способны решать задачу о коммивояжере для малого числа вершин ($n \approx 20$) за полиномиальное время. Поэтому для решения сложных задач о коммивояжере используются различные эвристики.

2.1. Выделим два основных критерия при решении задачи о коммивояжере. Первый оценивает скорость сходимости алгоритма. Второй оценивает близость получаемого решения к оптимальному (если оно известно) или же к лучшим стандартным решениям (бенчмаркам), полученным с помощью других эвристик (если оптимум не известен).

2.2. Рассмотрим классификацию алгоритмов решения задачи о коммивояжере. Эвристические алгоритмы для задачи о коммивояжере могут быть условно разбиты на два класса — класс, работающий только с одним решением (например, итерационные алгоритмы), и класс, работающий с множеством (популяцией) решений (например, генетические алгоритмы).

Первый класс алгоритмов обычно имеет более высокую скорость сходимости, но в общем случае не выходит из локальных оптимумов. Второй класс осуществляет параллельный поиск в нескольких точках пространства решений. Алгоритмы этого класса имеют больше шансов найти оптимальное решение, но обычно имеют временную сложность большего порядка.

2.3. Рассмотрим характерные для задачи о коммивояжере генетические операторы. Можно выделить два основных варианта представ-

ления решения задачи: кодирование в виде списка вершин (последовательность посещаемых городов) или кодирование в виде списка ребер, образующих гамильтонов цикл (список использованных путей из города в город).

Первый способ более простой. При этом в позиции i-й хромосомы находится i-й посещенный город (i-я вершина графа). Для графа G (рис. 5.12) приведем два возможных решения при таком методе кодирования:

1	2	3	4	5	6	7
1	4	2	3	5	7	6

1	2	3	4	5	6	7
6	4	2	3	1	5	7

Здесь в первой строке записаны номера вершин графа, а во второй строке — случайная подстановка, определяющая одно из возможных решений задачи о коммивояжере. Одной из характерных особенностей задачи о коммивояжере, как, впрочем, и большого ряда других задач, является ограничение на возможные комбинации значений в хромосоме. В случае задачи о коммивояжере имеем: один и тот же город не может быть посещен дважды и все города должны быть посещены.

Использование стандартных генетических операторов в алгоритмах решения задачи коммивояжера приводит к большому числу нереальных решений. Например, для графа G (рис. 5.12) возьмем из популяции $P = \{P_1, P_2, P_3, P_4, P_5\}$ две хромосомы (альтернативных решения) P_4, P_5:

$$P_4: \ 1\ 4\ 2\ 3\ 5\ 7\ 6$$
$$P_5: \ 6\ 4\ 2\ 3\ 1\ 5\ 7$$

Применим стандартный оператор кроссинговера. Пусть точка разрыва находится между третьим и четвертым геном в хромосоме. Тогда получим следующие два потомка:

$$P_6: \ 1\ 4\ 2\ |\ 3\ 1\ 5\ 7$$
$$P_7: \ 6\ 4\ 2\ |\ 3\ 5\ 7\ 6$$

Очевидно, что решения P_6, P_7 являются нереальными, так как в P_6 повторяется ген 1 и отсутствует ген 6, а в P_7 повторяется ген 6 и отсутствует ген 1, т. е. в решении задачи о коммивояжере содержатся города, посещаемые дважды и непосещаемые ни разу.

2.4. Интерес представляет матричное кодирование задачи о коммивояжере. Например, для хромосомы P_4 матрица задачи о коммивояжере запишется так:

$$R(P_4) = \left|\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right|.$$

Как видно из матрицы, единицами кодируются переходы между генами в хромосоме. Для хромосомы P_5 матрица задачи о коммивояжере примет следующий вид:

$$R(P_5) = \left|\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 6 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right|.$$

Тогда популяция хромосом будет состоять из набора аналогичных матриц. Такое кодирование ввиду большого числа нулей избыточное, хотя наглядно и удобно для преобразований. Применение стандартных генетических операторов для такой кодировки задачи о коммивояжере является неэффективным в виду большого количества получаемых нереальных решений. Использование вместо матриц списков связности позволит получить набор строительных блоков. На данном наборе на основе оператора сегрегации и оператора инверсии можно получать набор реальных потомков. Например, на основе $R(P_1)$ запишем набор строительных блоков: 1–4; 2–3; 3–5; 4–2; 5–6; 6–7; 7–1. Применив оператор сегрегации и оператор инверсии, получим новое реальное решение $P_8 : 4\,1\,7\,6\,5\,3\,2$ со значением целевой функции: $\texttt{Fitness}\,(\varphi(8)) = 17$.

2.5. Представление задачи коммивояжера в виде матрицы смежности дает возможность анализировать шаблоны, которые применимы для n-размерной оптимизации генетического алгоритма. Шаблоны, определенные в терминах единственной определенной позиции, соответствуют естественным строительным блокам, т. е. элементам для задачи TSP. Например, шаблон $(* * * * * * 7 *)$ есть множество всех изменений, в котором элемент 6, соответствующий гену 7, имеет место.

2.6. Для задачи коммивояжера существуют модифицированные операторы кроссинговера, позволяющие гарантированно получать реальные решения. К ним относятся: циклический, универсальный, частично соответствующий и «жадный» операторы кроссинговера, которые описаны выше.

3. «Жадные» стратегии применяются, когда искомый объект строится по технологии «снизу–вверх» (в нашем случае из строительных блоков). «Жадная» стратегия делает на каждом шаге «локально-оптимальный» выбор. Однако «жадная» стратегия не всегда дает оптимальное решение.

3.1. Приведем алгоритм построения оператора кроссинговера для задачи коммивояжера на основе «жадной» стратегии:

1. Для каждой пары хромосом случайным образом выбрать точку разрыва и в качестве номера стартовой вершины взять номер отмеченного гена в хромосоме.
2. Сравнить частичную стоимость путей, ведущих из текущих вершин в хромосомах родителях, и выбрать из них кратчайший.
3. Если выбранная таким образом вершина графа уже была включена в частичный путь, то взять случайную вершину из числа не просмотренных. Присвоить полученной вершине значение текущей.
4. При преждевременном образовании циклов выбрать другой кратчайший путь.
5. Повторять пункты 2 и 3, пока не будет построен гамильтонов цикл с квазиминимальной суммарной стоимостью ребер.
6. Конец работы алгоритма.

Решение-потомок в алгоритме формируется как последовательность вершин графа в том порядке, в котором они становились текущими.

3.2. «Жадные» операторы кроссинговера, основанные на знаниях о задаче, улучшают скорость сходимости, однако часто препятствуют выходу из локальных минимумов. Существует несколько способов избежать эту ситуацию: изменить архитектуру генетического алгоритма или же не помещать такие решения в популяцию. Для этой цели предложим несколько эвристик.

Эвристика 1 — предпочтение более невыгодного маршрута с точки зрения заданной целевой функции более выгодному с определенной вероятностью.

Эвристика 2 — выбор приблизительно пятидесяти процентов генетического материала от каждого из родителей (альтернативных решений).

Когда путь оказывается тупиковым, производится случайный или направленный выбор нерассмотренной вершины.

Опишем модифицированный «жадный» операторы кроссинговера:

1. Случайным образом выбрать стартовую вершину в первом родителе.
2. Текущий родитель — первый родитель.
3. Основной цикл оператора определить следующим образом. Выбрать, какая из соседних вершин (соседними являются вершины, расположенные в хромосоме справа и слева от рассматриваемой) ближе к текущей. Ближайшая из них, не вошедшая в путь, становится новой текущей вершиной. Если только одна из двух соседних вершин не вошла в путь, то она становится текущей. Если обе соседние вершины являются посещенными (тупиковая ситуация), то выбрать ближайшую вершину из числа не рассмотренных. Тогда в качестве текущего родителя выбрать другого родителя.

4. Повторять цикл оператора, пока все вершины не будут посещены. Потомок формируется как последовательность рассмотренных вершин.
5. Конец работы алгоритма.

3.3. Покажем пример работы алгоритма для графа G (рис. 5.12). Пусть популяция состоит из двух хромосом (P_9, P_{10}):

$$P_9: \ 1\ 4\ 2\ 3\ 5\ 7\ 6$$
$$P_{10}: \ 6\ 3\ 5\ 4\ 2\ 1\ 7$$

Для P_9 значение целевой функции: $\texttt{Fitness}(\varphi(9)) = 22$, а для P_{10} значение целевой функции: $\texttt{Fitness}(\varphi(10)) = 15$. Стрелка над одной из вершин означает, что данная вершина является текущей. В примере использован 50 % «жадный» оператор кроссинговера. На этапе 1 вершина 5 выбрана стартовой и является текущей для обоих родителей:

				↓		
1	4	2	3	5	7	6
		↓				
6	3	5	4	2	1	7

Согласно алгоритму, следующей вершиной в путь выбирается вершина 3, соседняя вершине 5. Получаем кратчайший путь (5–3):

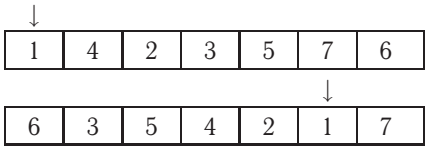
			↓			
1	4	2	3	5	7	6
	↓					
6	3	5	4	2	1	7

На следующем шаге в путь включается вершина 6, так как путь (5–3–6) имеет лучшее значение целевой функции, чем путь (5–3–7). Далее в путь включается вершина 7 и частичный путь имеет вид (5–3–6–7):

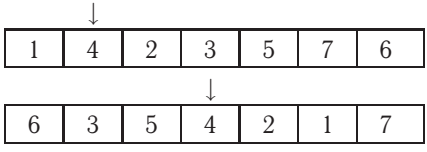
						↓
1	4	2	3	5	7	6
↓						
6	3	5	4	2	1	7
					↓	
1	4	2	3	5	7	6
						↓
6	3	5	4	2	1	7

Согласно алгоритму, далее в путь выбирается вершина 1. Это происходит из-за того, что в правой половине первого родителя рассмотрены

все вершины, а для второго родителя — вершина 1, соседняя с вершиной 7, и путь (5–3–6–7–1) кратчайший:



Далее в путь включается вершина 4, соседняя вершине 1 в первом текущем родителе. Тогда имеем путь (5–3–6–7–1–4).



Осталась не рассмотренной вершина 2, которая автоматически включается в путь. Тогда имеем альтернативное решение (потомок) P_{11} : 5 3 6 7 1 4 2 со значением целевой функции: $\texttt{Fitness}\,(\varphi(11)) = 26$. В связи с тем, что потомок P_{11} имеет значение целевой функции, хуже чем у родителей P_9, P_{10}, он в популяцию не включается. Снова в текущем родителе P_9 выбираем текущую вершину 7 и после нескольких шагов получаем хромосому-потомок P_{12} : 7 6 5 4 3 2 1 со значением целевой функции $\texttt{Fitness}\,(\varphi(12)) = 7$, что соответствует глобальному оптимуму. Такие операторы не всегда позволяют получать глобальный оптимум, поэтому необходимо использовать множество разработанных генетических операторов и нестандартные архитектуры генетического поиска.

На рис. 5.13 приведена структурная схема алгоритма решения задачи коммивояжера.

Здесь используется блок локального поиска, который позволяет получать локальные экстремумы в задаче коммивояжера. Все генетические операторы ориентированы на использование знаний о решаемой задаче. Использование генетических операторов варьируется в зависимости от конкретных особенностей графовой модели. Блок редукции оставляет размер популяции постоянным для каждой генерации.

3.4. После достижения локального оптимума в популяции наблюдается большое число одинаковых маршрутов. В этой связи может наступить преждевременная сходимость. Эта проблема тесно связана с потерей разнообразия в популяции. Одним из источников однообразия является появление «суперхромосом», которые в течение многих поколений доминируют в ней. Имеется ряд путей частичного решения этой проблемы, например, изменение алгоритма селекции или же модификация генетического оператора рекомбинации.

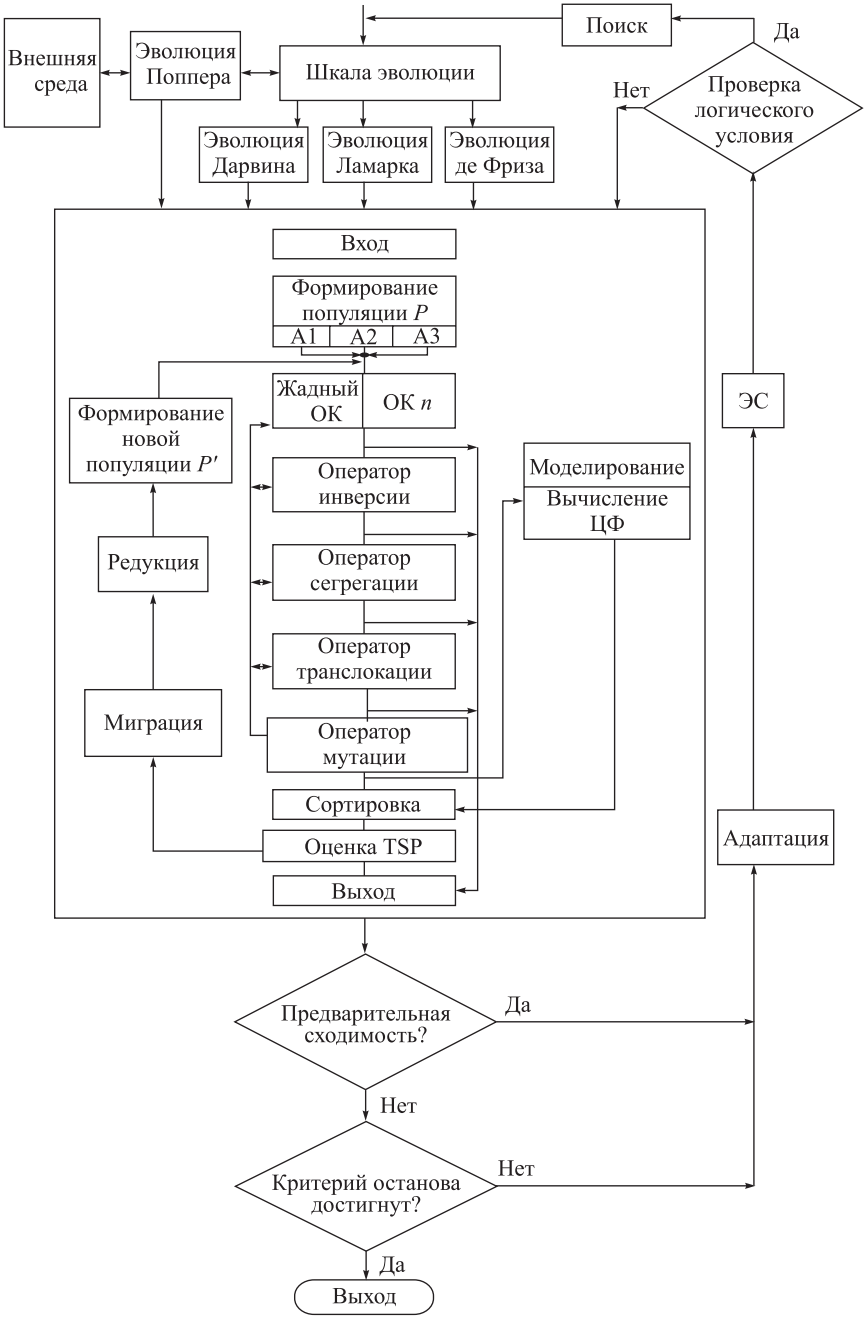


Рис. 5.13. Структурная схема алгоритма решения задачи коммивояжера

При попадании в локальный оптимум предлагается выполнять следующие эвристики:

Эвристика 1. Из всех маршрутов, имеющих одинаковую длину, только один остается неизменным. Над всеми другими выполняются модифицированные генетические операторы;

Эвристика 2. Не меняется только лучший маршрут. Ко всем другим применяются генетические операторы.

Отметим, что такие генетические операторы и эвристики работают не все время, а только в ситуациях, когда однообразие в популяции достигает высокого уровня. Совместная эволюция регламентирует и управляет таким процессом генетического поиска. Отметим, что отличительной особенностью рассмотренного генетического алгоритма является способность хорошо работать на популяциях с малым числом хромосом, что уменьшает временную сложность алгоритма.

5.3. Задачи раскраски, построения клик и независимых множеств графов

1. Рассмотрим задачу раскраски графа. ***Раскраской вершин графа*** $G = (X, U)$ называется разбиение множества вершин X на L непересекающихся классов (подмножеств):

$$X_1, X_2, \ldots, X_L; \quad X = \bigcup_{i=1}^{L} X_i; \quad X_i \bigcap X_j = \varnothing; \quad i, j = 1, 2, \ldots, L,$$

таких, что внутри каждого подмножества X_i не должно содержаться смежных вершин. Если каждому подмножеству X_i поставить в соответствие определенный цвет, то вершины внутри этого подмножества можно окрасить в один цвет, а вершины другого подмножества X_j в другой цвет и так далее до раскраски всех подмножеств. В этом случае граф называется L-раскрашиваемым.

Основной стратегией для задач раскраски графа являются последовательные и «жадные» эвристики, которые позволяют получать результаты с локальным оптимумом.

2. Последовательная эвристика заключается в следующем. Сначала составим упорядоченный в порядке убывания степеней вершин список. Перваю вершину удаляем из списка и окрашиваем в цвет 1. Просматривая список, в цвет 1 раскрашиваем и удаляем из него все вершины, несмежные с первой выбранной и между собой. Далее выбираем вторую вершину из списка, ее окрашиваем в цвет 2 и удаляем. Процесс продолжается аналогично, пока не будут окрашены все вершины.

Запишем основные «жадные» эвристики для раскраски.

Эвристика 1. «Первые наибольшие». Вершины графа упорядочиваются по убыванию локальных степеней. Затем по порядку каждая вершина после анализа связности раскрашивается в возможный цвет.

Эвристика 2. «Наименьшие последние». Построение порядка из хаоса производится следующим образом. Выбирается вершина графа с наименьшей локальной степенью и удаляется из графа вместе с инцидентными ребрами. В оставшемся графе повторяется та же процедура. Вершины графа раскрашиваются в минимально возможный для этой процедуры цвет в порядке, обратном порядку удаления.

Эвристика 3. «Разбиение по смежности». В начале алгоритма выбирается вершина графа максимальной степени и раскрашивается в цвет 1. Затем все нераскрашенные вершины разбиваются на два подмножества: X_1 — смежные с раскрашенными и X_2 — несмежные с раскрашенными. Выбор очередной вершины для раскраски в минимально возможный цвет осуществляется по наибольшей степени в X_1. Далее процесс продолжается аналогично до заполнения цвета 1. После этого вершины, раскрашенные цветом 1, удаляются, и процедура продолжается до полной раскраски графа.

Эвристика 4. «Раскраска ранее упорядоченных вершин». Для вершины x_i выбирается цвет k, если и только если среди вершин, смежных с x_i вершиной, существуют уже раскрашенные вершины цветов $c = 1, 2, \ldots, k-1, k+1, k+2, \ldots$ и нет вершин, помеченных цветом k.

Эвристика 5. «Специальная раскраска в предписанные цвета». Предварительно для каждой вершины задан набор цветов, в которые она может быть раскрашена. Упорядочение вершин производится следующим образом. Определяются вершины с минимальным числом $W = \rho(x_i) + z_i/C$, где $\rho(x_i)$ — локальная степень вершины $x_i \in X$ графа $G = (X, U)$, z_i — число запрещенных цветов для данной вершины, C — общее число цветов. Вершина x_i с минимальным значением W удаляется из графа вместе с ребрами. Для оставшегося графа повторяется такая же процедура, и так далее. Затем вершины красятся в возможный цвет в порядке, обратном порядку удаления.

Запишем простой алгоритм на основе одной из рассмотренных «жадных» эвристик:

1. Упорядочить вершины по убыванию локальных степеней.
2. Каждой вершине упорядоченного списка назначить первый имеющийся цвет, если это возможно.
3. Шаг 2 продолжать до полной раскраски графа.
4. Конец работы алгоритма.

Алгоритм на основе такой эвристики является «жадным», так как он раскрашивает граф последовательно вершину за вершиной, присваивая каждой вершине цвет, если это возможно, не учитывая глобальных последствий такой раскраски. «Жадный» алгоритм задачи раскраски графа быстр (временная сложность алгоритма $O(n)$), так как он рассматривает лишь один из возможных вариантов раскраски каждой вершины.

Рассмотрим, например, задачу раскраски графа (рис. 5.14). Упорядочиваем вершины следующим образом: $\langle 9\,7\,1\,8\,6\,3\,2\,5\,4 \rangle$. Сначала раскра-

шивается вершина 9, алгоритм оставляет вершину 7 не раскрашенной, так как она соединена с 9, а вершина 9 уже раскрашена. По той же причине пропускаем вершину 8. Далее закрашиваем вершину 1 и пропускаем остальные. Следовательно, $X_1 = \{9, 1\}$. В итоге получим $X_2 = \{7, 6, 5\}$, $X_3 = \{8, 3, 4\}$, $X_4 = \{2\}$. В итоге G (рис. 5.14) раскрашен 4 цветами.

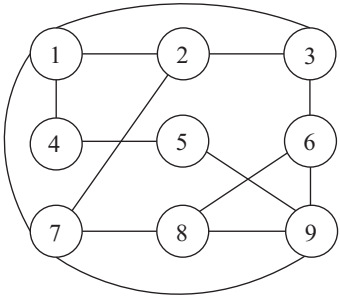


Рис. 5.14. Граф G

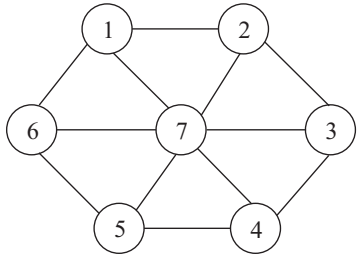


Рис. 5.15. Колесо W_7

Рассмотрим второй граф, изображенный на рис. 5.15. Это колесо W_7 содержит 7 вершин. Первые шесть вершин расположены в виде кольца, а седьмая вершина расположена в центре кольца и соединена со всеми остальными вершинами. «Жадная» эвристика из хаоса вершин графа создает следующий порядок: $\langle 7\,6\,5\,4\,3\,2\,1\rangle$. Алгоритм раскрасит вершины следующим образом: $X_1 = \{7\}$; $X_2 = \{6, 4, 2\}$ и $X_3 = \{5, 3, 1\}$. Получим, что колесо W_7 (рис. 5.15) раскрашено 3 цветами. Отметим, если в графе, аналогично колесу, имеется одна вершина, смежная всем остальным, то она раскрашивается одним цветом и удаляется из упорядоченного списка.

3. Так как «жадная» и последовательная эвристики раскраски графа достаточно быстрые, то предлагается совместить их с генетическим алгоритмом. Кодировка, используемая в «жадной» эвристике, основана на упорядочивании вершин графа. Она упорядочивает вершины по уменьшению значений локальных степеней вершин и затем декодирует эту последовательность, назначая каждой вершине по порядку первый реальный цвет, где реальность основана на использовании цветов при раскраске предыдущих вершин. Такой способ кодировки при решении проблемы раскраски графа упорядочивает вершины графа определенным способом, а затем декодирует их в соответствии с «жадной» эвристикой.

Здесь предлагается замена бинарной целевой функции на числовую или буквенную целевую функцию. Для определения такой целевой функции выбираем первый ген (вершину) в хромосоме и присваиваем ей первый реальный цвет, если это возможно. Каждая альтернатива учитывает предыдущие использованные цвета. Перестановка списка

элементов получается в результате упорядочивания элементов в списке (тривиальная перестановка просто оставляет предыдущий порядок). «Жадная» эвристика сортирует список вершин и передает результат в эвристику декодирования.

На следующем шаге алгоритма осуществляется выбор случайной перестановки или перестановки, основанной на знаниях о решаемой задаче. «Жадная» эвристика генерирует только одну хромосому, но эта хромосома имеет шанс оказаться лучше, чем средняя. В результате изменения порядка элементов в упорядоченном списке получим измененный список вершин графа. Применяя к полученным хромосомам генетические операторы, определяем значение целевой функции. В рассматриваемом алгоритме применяются элитная селекция и генетические операторы, описанные выше.

Запишем алгоритм реализации модифицированного универсального оператора кроссинговера. Даны родители 1 и 2, первый потомок получается следующим образом:

1. Сгенерировать бинарную строку-шаблон, количество элементов в которой совпадает с длиной хромосомы родителей.
2. Позиции хромосомы первого родителя, соответствующие единицам в строке шаблона, скопировать на следующий шаг.
3. Создать список генов из первого родителя, соответствующих нулю в бинарной строке.
4. Последовательно просматривая элементы второго родителя, произвести заполнение пустых позиций.
5. Объединяя частичные альтерантивные решения без повторений, получить первого потомка.
6. Конец работы алгоритма.

Построение второго потомка производится аналогичным образом.

Рассмотрим пример для графа, приведенного на рис. 5.14. Пусть альтернативные решения закодированы следующим образом:

$$\begin{array}{ll} P_1: & 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9 \\ P_2: & 9\ 7\ 5\ 3\ 1\ 8\ 6\ 4\ 2 \\ \text{БС}: & 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0 \end{array}$$

Результат после 2 шага:

$$\begin{array}{ll} P_1: & -\ 2\ -\ -\ -\ 6\ -\ -\ - \\ P_2: & 9\ -\ 5\ 3\ 1\ -\ 6\ 4\ 2 \end{array}$$

Окончательный результат:

$$\begin{array}{ll} P_3: & 9\ 2\ 5\ 3\ 1\ 6\ 7\ 4\ 8 \\ P_4: & 9\ 8\ 5\ 3\ 1\ 7\ 6\ 4\ 2 \end{array}$$

Здесь БС — бинарная строка, показывающая смежность соответствующих вершин в P_1 и P_2, Иначе говоря, 1 в БС показывает, что

вершины 5 и 9 (рис. 5.14) связаны между собой. Если смежных вершин в заданных родителях нет, то бинарная строка заполняется 0 и 1 случайным образом. Выполнив «жадное» декодирование, получим, что хромосома P_3 соответствует следующей раскраске графа: $X_1 = \{9, 2, 4\}$, $X_2 = \{5, 3, 7\}$, $X_3 = \{1, 6\}$ и $X_4 = \{8\}$. Для P_4 получим другую раскраску графа: $X_1 = \{9, 3, 4\}$, $X_2 = \{8, 5, 1\}$, $X_3 = \{7, 6\}$ и $X_4 = \{2\}$. Заметим, что основная проблема здесь — декодирование модифицированного универсального оператора кроссинговера. Эта операция имеет временную сложность $O(n^2)$, где n — размер хромосомы.

Опередилим модифицированный оператор мутации. Он выбирает часть родительской хромосомы и, произведя перестановки внутри нее, переносит эту часть в потомка, а остальную оставляет без изменений. Все генетические операторы такого вида имеют сложное декодирование. Опишем «жадную» стратегию с одновременным декодированием хромосом. Пусть для графа G (рис. 5.14) задана популяция $P = \{P_1, P_2, P_3, P_4\}$:

$$
\begin{array}{lccccccccc}
P_1: & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
P_2: & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\
P_3: & 1 & 3 & 5 & 7 & 9 & 2 & 4 & 6 & 8 \\
P_4: & 9 & 7 & 5 & 3 & 1 & 8 & 6 & 4 & 2
\end{array}
$$

В хромосоме P_1 выбирается ген (элемент альтернативного решения), соответствующий вершине с максимальной степенью. Если таких вершин несколько, то они анализируются последовательно в произвольном порядке. В первой хромосоме выбираем элемент 9. Согласно алгоритму, следующей выбирается вершина 1. Дальнейшие шаги алгоритма на всей популяции приводят к тупиковым ситуациям. Следовательно, определен первый цвет $X_1 = \{9, 1\}$. Элементы 9, 1 исключаются из всех хромосом популяции P. Далее в P_1 выбирается элемент 7. Согласно алгоритму получим второй цвет $X_2 = \{7, 6, 4\}$. Продолжая, аналогично найдем задачу раскраски графа (рис. 5.14), состоящую из четырех цветов $X_3 = \{8, 2, 5\}$ и $X_4 = \{3\}$. Данная методика позволяет сокращать размер хромосом (альтернативных решений) на каждом шаге и, соответственно, время раскраски графа.

4. Кроме генетических операторов необходимо также определить тип селекции. Для задачи раскраски графа применима модифицированная элитная селекция, которая реализуется по следующей схеме. Все хромосомы t-го поколения упорядочиваются в порядке убывания значений степеней вершин (приспособленности к внешней среде).

Структура гибридного генетического алгоритма, используемого для задачи раскраски графа, представлена на рис. 5.16.

В данной схеме используются идеи совместной эволюции, установление баланса, адаптации к внешней среде, активное взаимодействие с внешней средой, иерархическое управление генетическим поиском, локальный поиск решений и применение всех модифицированных гене-

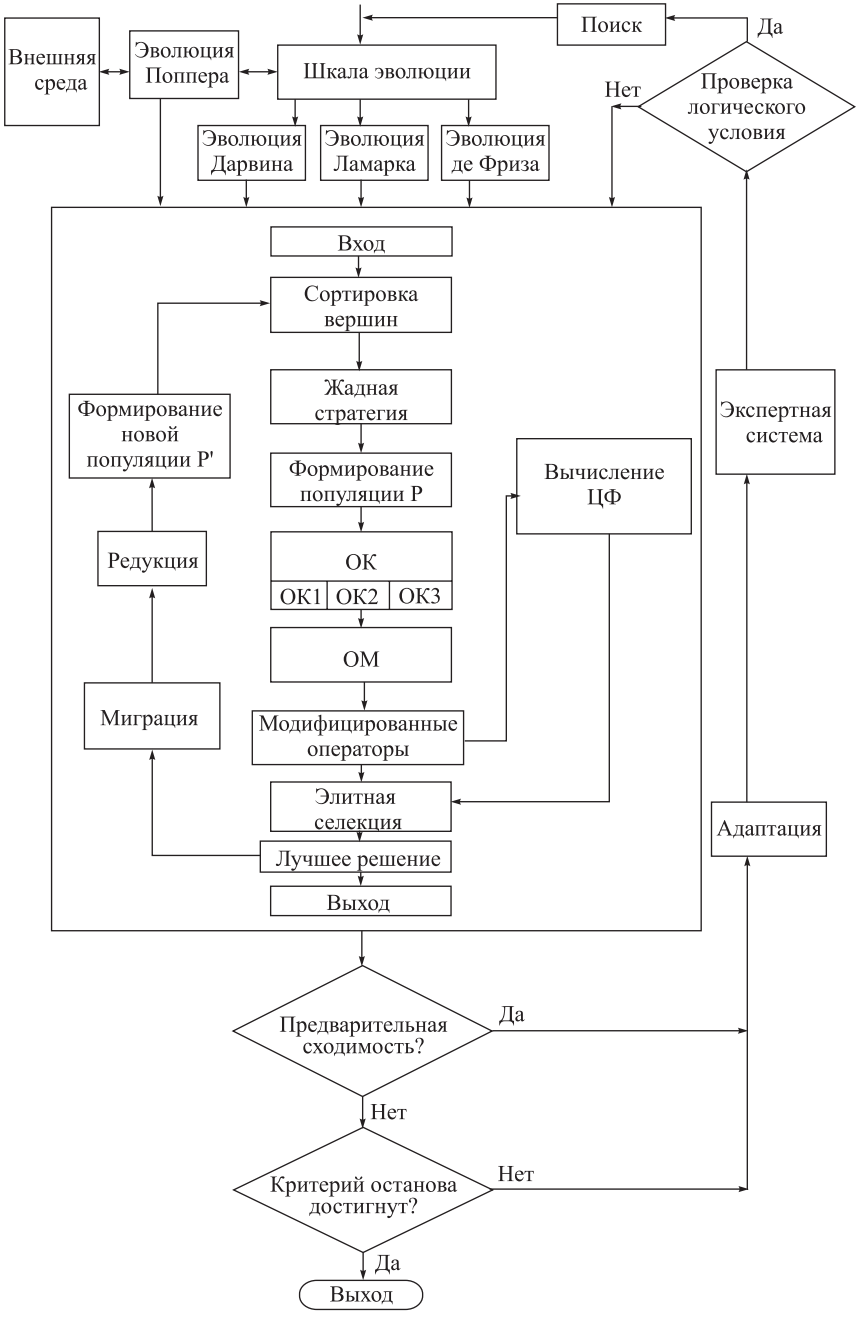


Рис. 5.16. Структура гибридного генетического алгоритма

тических операторов на основе «жадной» стратегии и поисковых методов. Временная сложность алгоритма такого класса лежит в пределах $O(n^2) - O(n^3)$, где n — длина хромосомы (альтернативного решения).

5. Задача раскраски графа тесно связана с построением независимых или внутренне устойчивых подмножеств, а также определением клик графа. Если две любые вершины подмножества X' графа $G = (X, U)$, $X' \subseteq X$, не смежны, то оно называется ***внутренне устойчивым***. Подмножество $\Psi_i \subseteq X$ графа $G = (X, U)$ называется ***максимальным внутренне устойчивым подмножеством*** или ***независимым***, если добавление к нему любой вершины $x_i \subseteq X$ делает его не внутренне устойчивым. Подмножество Ψ_i, будет независимым, если

$$\forall x_i \in \Psi_i (x_i \cap \Psi_i = \varnothing).$$

Независимые подмножества различаются по числу входящих в них элементов. В произвольном графе G можно выделить семейство всех независимых подмножеств вида $\Psi_i = \{\Psi_1, \Psi_2, \ldots, \Psi_s\}$ или его части. «Жадные» стратегии, используемые для задачи раскраски графа, также эффективно применяются для построения семейства независимых подмножеств. Например, построим семейство независимых подмножеств для графа G, представленного на рис. 5.17. Предварительно произведем упорядочивание вершин по возрастанию, начиная с вершины с наименьшей локальной степенью. Работу «жадного» оператора кроссинговера покажем на примере.

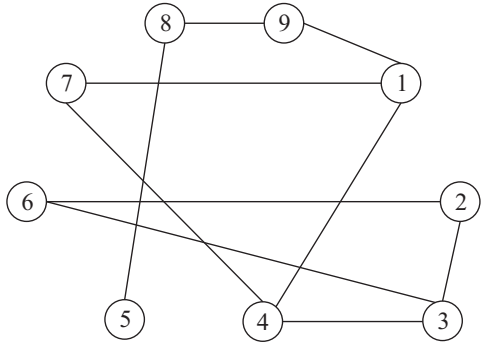


Рис. 5.17. Граф G для определения независимых подмножеств

Пусть для графа G (рис. 5.17) задана популяция $P = \{P_1, P_2, P_3, P_4, P_5\}$:

P_1 : 1 2 3 4 5 6 7 8 9
P_2 : 9 8 7 6 5 4 3 2 1
P_3 : 6 1 5 4 7 9 3 8 2
P_4 : 4 9 3 7 8 1 7 6 2
P_5 : 5 6 7 8 9 2 3 4 1

Работа алгоритма начинается с выбора в упорядоченной хромосоме P_5 первого элемента — 5. Далее выбираем вершину 6, так как она соседняя вершине 5 в хромосоме P_5 и не смежна ей. Аналогично выбираем 7 вершину в этой хромосоме. После этого, анализируя популяцию P, в хромосоме P_3 выбираем вершину 9. Дальнейший анализ популяции P показывает, что построено первое независимое подмножество $\Psi_1 = \{5, 6, 7, 9\}$.

Существует большое число стратегий построения независимых подмножеств. Одна из них состоит в построении независимых строительных блоков на две, три, четыре и так далее вершин с дальнейшим объединением этих строительных блоков в независимое подмножество. Другая стратегия (в ширину) предусматривает нахождение всех независимых подмножеств, в которые входит первая рассматриваемая вершина (в нашем случае вершина 5), с дальнейшим анализом других вершин на предмет создания новых независимых подмножеств. Третья стратегия (в глубину) предусматривает последовательный анализ всех вершин в упорядоченном списке (хромосома P_5) до получения семейства независимых подмножеств. Четвертая стратегия является комбинацией первых трех. Применяя четвертую стратегию для графа G (рис. 5.17), получим набор независимых подмножеств

$$\Psi_2 = \{6, 7, 8\}, \quad \Psi_3 = \{7, 8, 2\}, \quad \Psi_4 = \{8, 1, 3\}, \quad \Psi_5 = \{9, 2, 4, 5\},$$
$$\Psi_6 = \{1, 2, 5\}, \quad \Psi_7 = \{2, 5, 7, 9\}, \quad \Psi_8 = \{3, 5, 7, 9\}, \quad \Psi_9 = \{4, 5, 6, 9\}.$$

Очевидно, что данный набор не полный. Для получения полного семейства независимых подмножеств необходимо продолжить реализацию данных стратегий.

Описанная методика эффективно используется для задачи раскраски графа. Например, пусть Ψ_5 задает первый цвет $X_1 = \{9, 2, 4, 5\}$, Ψ_2 задает второй цвет $X_2 = \{6, 7, 8\}$, а Ψ_3 задает третий цвет $X_3 = \{1, 3\}$. Следовательно, граф G (рис. 5.17) раскрашен тремя красками. Временная сложность данного алгоритма лежит в пределах $O(n^2) - O(n^4)$.

6. В алгоритмах определения независимых множеств можно использовать методику кодирования, когда каждое решение представляет собой бинарную хромосому длины N, где N — число вершин исследуемого графа G. Каждому гену соответствует определенная вершина графа. Вершины упорядочиваются, например, в порядке возрастания их номеров: первому гену соответствует первая вершина графа, второму гену — вторая вершина и т. д. Если значение гена равно 1, то данная вершина включается в искомое подмножество, если ноль — то вершина не включается.

Например, пусть задана хромосома

1	2	3	4	5	6	7
1	0	1	0	1	1	0

Такая запись показывает, что независимое подмножество имеет вид: $X' = \{x_1, x_3, x_5, x_6\}$, так как гены 1, 3, 5, 6 рассматриваемой хромосомы равны единице.

Пусть для случайного графа на 7 вершин задана начальная популяция $P = \{P_1, P_2, P_3, P_4, P_5\}$:

$$\begin{array}{lcccccccc} P_1: & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ P_2: & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ P_3: & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ P_4: & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ P_5: & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}$$

Согласно описанному выше принципу кодирования единицы соответствуют наличию вершины в соответствующем подмножестве. Здесь хромосома P_3 соответствует независимому подмножеству.

Например, после первой генерации алгоритма популяция примет вид (нарисуйте граф!)

$$\begin{array}{lcccccccc} P'_1: & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ P'_2: & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ P'_3: & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ P'_4: & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ P'_5: & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{array}$$

Хромосомы P'_2 и P'_3 соответствуют независимым подмножествам.

6.1. Для оценки качества полученных решений применяется следующая процедура. В хромосоме (строке) выбирается первая позиция, и затем данная числовая последовательность просматривается слева направо. На каждом шаге проверяется выполнение заданного условия. Так, при построении независимых подмножеств таким условием является отсутствие общих ребер между рассматриваемыми вершинами. В результате выполнения данной процедуры будет сформировано одно независимое подмножество некоторой длины.

В некоторых случаях после создания начальной популяции решений и выделения в них независимых подмножеств производится переупорядочение всех хромосом. Вначале записывается последовательность чисел, соответствующая сформированному на базе данной хромосомы подмножеству, а затем — оставшиеся вершины. Точка кроссинговера выбирается случайно среди первых L_p позиций. Величина L_p соответствует мощности сформированного подмножества. Например, для произвольного графа G на 6 вершин можно сгенерировать некоторое решение:

6	3	2	4	5	1

Просматривая данную последовательность, можно например получить независимое подмножество $X' = \{x_6, x_2, x_1\}$. Переупорядочивая эту числовую последовательность, получим:

6	**2**	**1**	3	4	5

Цифры, выделенные жирным шрифтом, соответствуют независимому подмножеству, а остальные цифры оставшимся вершинам графа.

6.2. Приведем генетический алгоритм выделения независимого подмножества.

1. Ввести параметры алгоритма.
2. Сгенерировать начальную популяцию на основе стратегии дробовика.
3. Выбрать хромосомы-родителей на основе случайной селекции.
4. Реализовать оператор кроссинговера с заданной вероятностью.
5. Реализовать оператор мутации для каждой хромосомы с заданной вероятностью.
6. Реализовать оператор отбора для выбора лучших хромосом.
7. Конец работы алгоритма.

Покажем работу алгоритма на примере.

7. Для сокращения пространства поиска используется хромосома, длина которой равна наибольшему используемому подмножеству. Для этого используют формулы оценки нижней $l_{\min}$ и верхней $l_{\max}$ границы числа внутренней устойчивости.

Число генов в хромосоме равно верхней оценке числа внутренней устойчивости $l_{\max}$. Значением гена является номер вершины в графе G. Заполненная часть хромосомы l представляет собой внутренне устойчивое или независимое подмножество, $l \in [l_{\min}, l_{\max}]$. Так как l может быть меньше длины хромосомы, то оставшаяся часть хромосомы заполняется нулями.

Целевой функцией задачи определения независимых подмножеств является l, а целью оптимизации — максимизация функции, т. е.

$$F(P_i) = l_{Pi}, \quad F(P_i) \to \max.$$

Начальная популяция хромосом представляет собой множество внутренне устойчивых подмножеств, некоторые из которых могут быть и независимыми. Сначала случайным образом формируется множество вершин, мощность которого выбирается случайно в пределах $[l_{\min}, l_{\max}]$. Проверяется, является ли это множество внутренне устойчивым, если да, то хромосома записывается в начальную популяцию, если нет, то формируется новое множество вершин. Процесс продолжается до тех пор, пока не будет сформировано заданное число хромосом.

Приведем схему работы модифицированного оператора упорядочивающего одноточечного кроссинговера.

Из двух родительских хромосом выбирается та, длина заполненной части (l_1) которой меньше. Случайным образом выбирается точка скрещивания в пределах $[1, l_1]$. Далее в хромосому первого потомка копируется хромосома первого родителя до точки кроссинговера, а гены потомка, расположенные правее точки скрещивания, записываются в последовательности, соответствующей второму родителю. При этом второй родитель просматривается от начала до конца, слева направо, и элементы, которых не хватает в потомке, добавляются, начиная от точки кроссинговера, по порядку. Для создания второго потомка применяется обратный порядок действий.

Рассмотрим пример работы такого кроссинговера:

Родитель 1:	12	10	6	3	0	0	0	0	0
Родитель 2:	1	7	10	13	9	4	0	0	0
Потомок 1:	12	10	1	7	13	9	4	0	0
Потомок 2:	1	7	12	10	6	3	0	0	0

Опишем работу модифицированного оператора мутации. Пусть заполненная часть хромосомы-родителя равна l_1, а нулевая l_0. Случайным образом сформируем множество вершин A, не принадлежащих родительской хромосоме, $|P_i| \in [1, l_0]$. Допишем эти вершины в хромосому. Зададим случайное число k, $k \in [0, l_1]$, и удалим k генов от начала хромосомы. Полученное множество вершин записывается в хромосому потомка.

Рассмотрим пример работы предложенного оператора мутации:

Родитель:	12	10	6	3	0	0	0	0	0
После добавления генов:	12	10	6	3	7	9	11	0	0
Потомок :	6	3	7	9	11	0	0	0	0

Поясним работу алгоритма.

На первом шаге алгоритма вводятся начальные данные:

- исходный граф $G = (X, U)$;
- параметры генетического алгоритма (вероятности кроссинговера, мутации, элитного и равновероятного отбора; размер популяции хромосом; количество генераций генетического алгоритма).

Далее рассчитываются верхняя и нижняя оценки числа внутренней устойчивости и формируется начальная популяция. После этого начинается итеративный процесс применения генетических операторов к исходной популяции. На каждой итерации с заданной вероятностью к хромосомам текущей популяции применяются операторы кроссинговера и мутации. Затем рассчитываются целевые функции хромосом, и производится их проверка на независимость. Независимые подмножества заносятся в отдельный список. После этого в соответствии с заданной вероятностью производится элитный или равновероятный отбор хромосом в новую популяцию. Процесс продолжается до тех пор, пока не будет выполнено условие останова — заданное число генераций.

8. Заметим, что обратной для построения независимого подмножества является задача построения клик графа. Подмножество из максимального числа смежных между собой вершин в графе называется ***кликой***. В графе можно выделить некоторое семейство клик: $Q = \{Q_1, Q_2, \ldots, Q_z\}$. Очевидно, что стратегии, описанные выше, могут быть применены для определения семейства клик. Покажем на примере графа G (рис. 5.18) упрощенную схему работы генетического алгоритма определения клик графа.

Пусть для графа G (рис. 5.18) задана популяция $P = \{P_1, P_2, P_3, P_4, P_5\}$:

P_1:	1	2	3	4	5	6	7	8	9	10	11	12
P_2:	12	11	10	9	8	7	6	5	4	3	2	1
P_3:	2	4	6	8	10	12	1	3	5	7	9	11
P_4:	1	5	9	2	6	10	3	7	8	4	11	12
P_5:	1	10	12	6	2	9	11	4	8	7	3	5

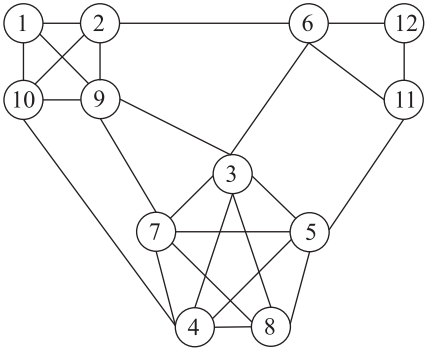


Рис. 5.18. Граф G для определения клик

Применим следующую схему поиска, основанную на «жадной» стратегии:

1) 1.
2) 1, 2 — образовано полное подмножество на две вершины.
3) 1, 2, 3 ♦ (тупик).
4) 1, 3 ♦.
5) 1, 5 ♦.
6) 1, 10, 11 ♦.
7) 1, 10, 9, 11 ♦.
8) 1, 10, 9, 8 ♦.
9) 1, 10, 9, 2, 3 ♦.
10) 1, 10, 9, 2, 4 ♦.
11) 1, 10, 9, 2, 6 ♦.

Построена первая клика $Q_1 = \{1, 10, 9, 2\}$. Элементы, соответствующие вершинам Q_1, удаляются из всех хромосом популяции. Получаем новую популяцию с хромосомами меньшей длины:

$$
\begin{array}{rcccccccc}
P_6: & 3 & 4 & 5 & 6 & 7 & 8 & 11 & 12 \\
P_7: & 12 & 11 & 8 & 7 & 6 & 5 & 4 & 3 \\
P_8: & 4 & 6 & 8 & 12 & 3 & 5 & 7 & 11 \\
P_9: & 5 & 6 & 3 & 7 & 8 & 4 & 11 & 12 \\
P_{10}: & 12 & 6 & 11 & 4 & 8 & 7 & 3 & 5
\end{array}
$$

Продолжая далее:

1) 3.
2) 3, 4.
3) 3, 4, 5.
4) 3, 4, 5, 6 ♦.
5) 3, 4, 5, 12 ♦.
6) 3, 4, 5, 7, 8, 11 ♦.
7) 3, 4, 5, 7, 8, 6 ♦.
8) 3, 4, 5, 7, 8, 12 ♦.

Построена вторая клика $Q_2 = \{3, 4, 5, 7, 8\}$. Далее, продолжая аналогично, построим третью клику $Q_3 = \{6, 11, 12\}$. Отметим, что за 21 элементарный шаг на основе «жадной» модифицированной стратегии генетического поиска получены основные три клики графа. Временная сложность алгоритма в лучшем случае $O(n)$, в самом худшем случае $O(n!)$. В среднем для реальных графов $O(n^2) \div O(n^3)$.

5.4. Определение планарности графов на основе генетического поиска

1. Проблема определения планарности и получения плоской укладки графа относится к числу известных задач теории графов.

Граф $G = (X, U)$ называют ***плоским***, если существует такое изображение на плоскости его вершин и ребер, что каждая вершина $x_i \in X$ изображается на плоскости отдельной точкой, при этом ребра $u_j \in U$ простой кривой, имеющей концевые точки (x_i, x_k), пересекаются толь-

ко в общих точках, т. е. в вершинах. Здесь и далее под термином граф понимается связный неориентированный граф.

Граф называется ***планарным***, если он изоморфен плоскому, т. е. существует возможность получения плоской укладки такого графа. Область плоскости, ограниченная ребрами плоского графа и не содержащая внутри себя ни вершин, ни ребер, называется ***гранью***. Известная формула Эйлера связывает число вершин и ребер плоского графа с числом его граней: $n - m + f = 2$, где n — число вершин, m — число ребер графа, f — число граней графа. Из указанной формулы был выведен ряд следствий.

Следствие 1. *В любом простом планарном графе существует вершина, степень которой не больше пяти.*

Следствие 2. *Каждый планарный граф G с $n \geqslant 4$ вершинами имеет по крайней мере четыре вершины со степенями, не превышающими* 5.

Следствие 3. *Если G — связный простой планарный граф с $n \geqslant 3$ вершинами и m ребрами, то $m \leqslant 3n - 6$.*

Приведенные следствия определяют зависимость планарности графа от числа его вершин и ребер и задают границы интервала по числу ребер, для которого необходимо проводить дополнительные исследования, чтобы получить достоверный ответ на вопрос, планарен ли исследуемый граф.

Все графы можно условно разбить на три непересекающихся подмножества: G'_1, G'_2 и G'_3. В первое подмножество G'_1 входят заведомо планарные графы, для которых справедливо неравенство $m \leqslant n + 2$. Второе подмножество G'_2 образуют заведомо непланарные графы, для которых справедливо неравенство $m > 3(n - 2)$. Третье подмножество G'_3 составляют графы, число ребер которых находится в пределах интервала $(n + 2) < m \leqslant 3(n - 2)$. Для этих графов требуются дополнительные исследования, поскольку каждый из них может быть как планарным, так и не планарным.

Следствие 4. *Графы K_5 и $K_{3,3}$ не являются планарными.*

Здесь K_5 — это полный граф на пять вершин, а $K_{3,3}$ — двудольный однородный граф с локальной степенью вершин, равной трем (рис. 5.19).

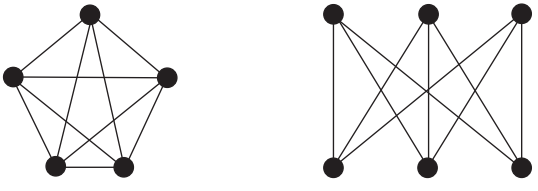


Рис. 5.19. Графы K_5 и $K_{3,3}$

Критерий Понтрягина–Куратовского. *Граф планарен тогда и только тогда, когда он не содержит подграфа, гомеоморфного* K_5 *или* $K_{3,3}$.

Данный критерий является наиболее распространенным. Смысл его можно пояснить на примере. Пусть задан граф $G = (X, U)$, не имеющий петель и кратных ребер. *Подразбиением* ребра $u = (x_i, x_j)$ называется замена данного ребра двумя ребрами $u_1 = (x_i, x_k)$ и $u_2 = (x_k, x_j)$ с введением промежуточной вершины x_k. Два графа называются *гомеоморфными*, если они имеют изоморфные подразбиения. Примеры графов, гомеоморфных графам K_5 и $K_{3,3}$, показаны на рис. 5.20.

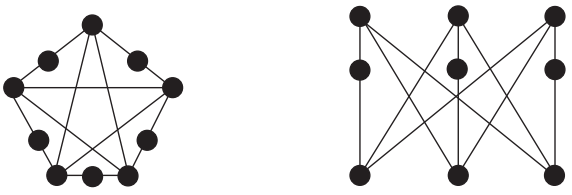


Рис. 5.20. Графы, гомеоморфные графам K_5 и $K_{3,3}$

Существуют и другие критерии планарности.

Критерий Харари–Татта. *Граф планарен тогда и только тогда, когда у него нет подграфов, стягиваемых к* K_5 *и* $K_{3,3}$.

Критерий Уитни. *Граф* G *планарный тогда и только тогда, когда он имеет комбинаторно двойственный граф* G^*.

Уитни связал планарность графов с существованием двойственных графов. Для построения *геометрически двойственного* графа G^* каждая грань исходного графа G, включая внешнюю, преобразуется в вершину графа G^*, при этом вершины графа G^* будут смежными, если были смежными, т. е. имели общее ребро, соответствующие грани графа G.

Критерий Мак-Лейна. *Граф* G *планарен тогда и только тогда, когда каждый его блок, содержащий, по крайней мере, три вершины, обладает таким базисом циклов* $Z_1, Z_2, \ldots, Z_m$ *и таким дополнительным циклом* Z_0, *что любое ребро блока принадлежит точно двум из этих* $m + 1$ *циклов*.

Этот критерий основан на рассмотрении циклической структуры графа. Как следует из формулы Эйлера, все внутренние грани двусвязного плоского графа G образуют базис циклов $Z_1, Z_2, \ldots, Z_m$, где m — циклический ранг графа G. Кроме того, имеется также Z_0 — внешний простой цикл графа G. Тогда любое ребро такого графа G будет принадлежать точно двум из $m + 1$ циклов Z_i.

2. Практическое использование перечисленных критериев затруднено из-за необходимости полного перебора для содержательного рас-

смотрения графа. Поэтому были разработаны эвристические методы определения планарности. Условно их можно разделить на три класса: циклические, матричные и комбинированные. Рассмотрим кратко идею каждого из перечисленных подходов.

Циклический метод определения планарности заключается в выделении в заданном графе произвольных циклов, в результате чего граф разбивается на подграфы, каждый из которых проверяют на планарность, уменьшая таким образом размерность задачи. Планарность исходного графа определяется из планарности подграфов, полученных в результате выделения циклов. Основной недостаток этих методов заключается в том, что задача выделения необходимых циклов сложна и требует значительных затрат времени.

Матричные методы определения планарности используют некоторые специальные матрицы графа, например, матрицу линейно независимых циклов. При этом отправной точкой является критерий Мак-Лейна, отмечающий, что граф планарный, если в матрице всех его возможных циклов можно выделить подматрицу, содержащую $m - n + 2$ строк и m столбцов, причем каждый столбец содержит строго две единицы. Однако такой вариант подразумевает большие затраты времени, особенно если исследуемый граф оказывается непланарным.

Методы и алгоритмы определения планарности можно разделить на две группы, каждая из которых состоит их нескольких подклассов (рис. 5.21).

Работу матричного алгоритма определения планарности проиллюстрируем на примере. Пусть задан граф G, изображенный на рис. 5.22.

Рис. 5.21. Классификация методов определения планарности

Граф имеет $n = 6$ вершин, $m = 9$ ребер, а число строк, которое необходимо выделить из матрицы циклов B_a для установления планарности графа, будет равно $m - n + 2 = 9 - 6 + 2 = 5$.

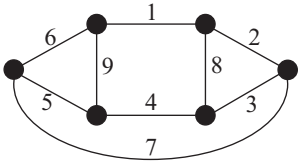


Рис. 5.22. Исходный граф G

3. Сформируем реберно-ориентированную матрицу B_{am}, вектор множества ребер M и вектор комбинаций циклов C_k (рис. 5.23). В векторе C_k выбирается первая комбинация $C_1 = [1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0]$ и находится алгебраическое произведение векторов: $C_1 M = 3 + 3 + 4 + 4 + 4 = 18$. На следующем шаге полученное значение $C_1 M = p$ сравнивается с числом $2m$. В случае, если $p < 2m$, алгоритм генерирует новую комбинацию циклов и процесс повторяется. Если $p > 2m$, то исследуемый граф непланарный. В случае же, когда $p = 2m$, как, например, в рассматриваемом примере, алгоритм продолжает работу.

$$B_{\text{am}} = \begin{bmatrix} 0\,1\,1\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1\,1\,0\,0\,1 \\ 0\,0\,1\,1\,1\,0\,1\,0\,0 \\ 1\,1\,0\,0\,0\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,0\,1 \\ 1\,0\,0\,1\,1\,1\,0\,1\,0 \\ 1\,1\,0\,0\,1\,0\,1\,0\,1 \\ 0\,1\,0\,1\,1\,0\,1\,1\,0 \\ 1\,0\,1\,0\,0\,1\,1\,1\,0 \\ 0\,0\,1\,1\,0\,1\,1\,0\,1 \\ 1\,1\,1\,1\,1\,1\,0\,0\,0 \\ 1\,0\,1\,0\,1\,0\,1\,1\,1 \\ 0\,1\,0\,1\,0\,1\,1\,1\,1 \end{bmatrix}; \qquad M = \begin{bmatrix} 3 \\ 3 \\ 4 \\ 4 \\ 4 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 6 \\ 6 \\ 6 \end{bmatrix}.$$

Рис. 5.23. Реберно-ориентированная матрица и вектор множества ребер

$$B_e = \begin{bmatrix} 0\,1\,1\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1\,1\,0\,0\,1 \\ 0\,0\,1\,1\,1\,0\,1\,0\,0 \\ 1\,1\,0\,0\,0\,1\,1\,0\,0 \\ 1\,0\,0\,1\,0\,0\,0\,1\,1 \end{bmatrix}.$$

Рис. 5.24. Подматрица B_e

Если в результате удалось сформировать подматрицу B_e (рис. 5.24), отвечающую критерию Мак-Лейна, то граф планарный, в противном случае после генерации новой комбинации циклов процесс повторяется.

Основная проблема рассмотренного алгоритма — это генерация и хранение необходимых для работы алгоритма массивов данных (матрицы инцидентности, циклов, реберно-ориентированной матрицы и так далее).

К числу достоинств алгоритма можно отнести возможность гарантированного определения планарности.

4. Циклические алгоритмы начинают свою работу с выделения в исследуемом графе $G = (X, U)$ некоторого простого, т. е. не имеющего повторяющихся вершин, цикла произвольной длины. Предварительно

проводится исследование заданного графа на предмет удаления висячих вершин и инцидентных им ребер, а также стягивания вершин, имеющих локальную степень, равную двум. Кроме того, необходимо проверить, является ли исходный граф связным, а также определить, не является ли рассматриваемый граф заведомо планарным либо заведомо непланарным. После выполнения всех операций производится факторизация связных компонент, т. е. стягивание всех вершин и ребер, принадлежащих компоненте, в одну вершину. Затем каждая образовавшаяся компонента, включая цикл, проверяется на возможность построения плоской укладки. В графе находится произвольный цикл, после удаления которого граф распадается на отдельные фрагменты. Затем производится попытка расположить каждый полученный фрагмент на плоскость во внешней либо внутренней областях первоначального цикла. Далее укладки фрагментов, по возможности, объединяют, получая плоскую укладку всего графа. Алгоритм основан на идеях поиска в глубину и имеет временную сложность $O(n \log n)$.

Рассмотрим механизм использования наследственной информации (рис. 5.25) на примере задачи минимизации пересечений. Для этого в заданном графе выделяем произвольным образом некоторый цикл, разбивая таким образом всю плоскость на две полуплоскости, причем все ребра, инцидентные вершинам цикла, располагаются либо внутри, либо снаружи цикла. В итоге получаем начальную «родительскую» пару хромосом, каждая из которых несет информацию о расположении ребер, при этом целевой функцией будет число пересечений ребер в данной полуплоскости.

5. При решении задачи минимизации пересечений возможен вариант, когда из одной полуплоскости в другую перемещают не отдельные ребра, а последовательность ребер, расположенных в строго определенном порядке. В этом случае хромосома представляет собой закодированную в определенной последовательности цепочку ребер, и каждый ее разряд есть звено цепи. Следовательно, применение стандартных операторов может привести к разрыву цепи и получению нереального решения.

Для применения универсального оператора кроссинговера к подобной хромосоме необходимо предварительно сформировать маску таким образом, чтобы при обмене ребрами во время выполнения оператора кроссинговера изменение внутренней структуры цепочек ребер не приводило к образованию разрывов.

Рассмотрим порядок выполнения оператора мутации для определения планарности графа. Например, в ходе решения задачи получена векторная хромосома, каждый ген которой представляет собой закодированную последовательность ребер. Для выполнения операции мутации случайным образом выбираем ген:

$\{1,2,3\}$	$\{4,5,6\}$	$\{7,8,9\}$	$\{10,11,12\}$

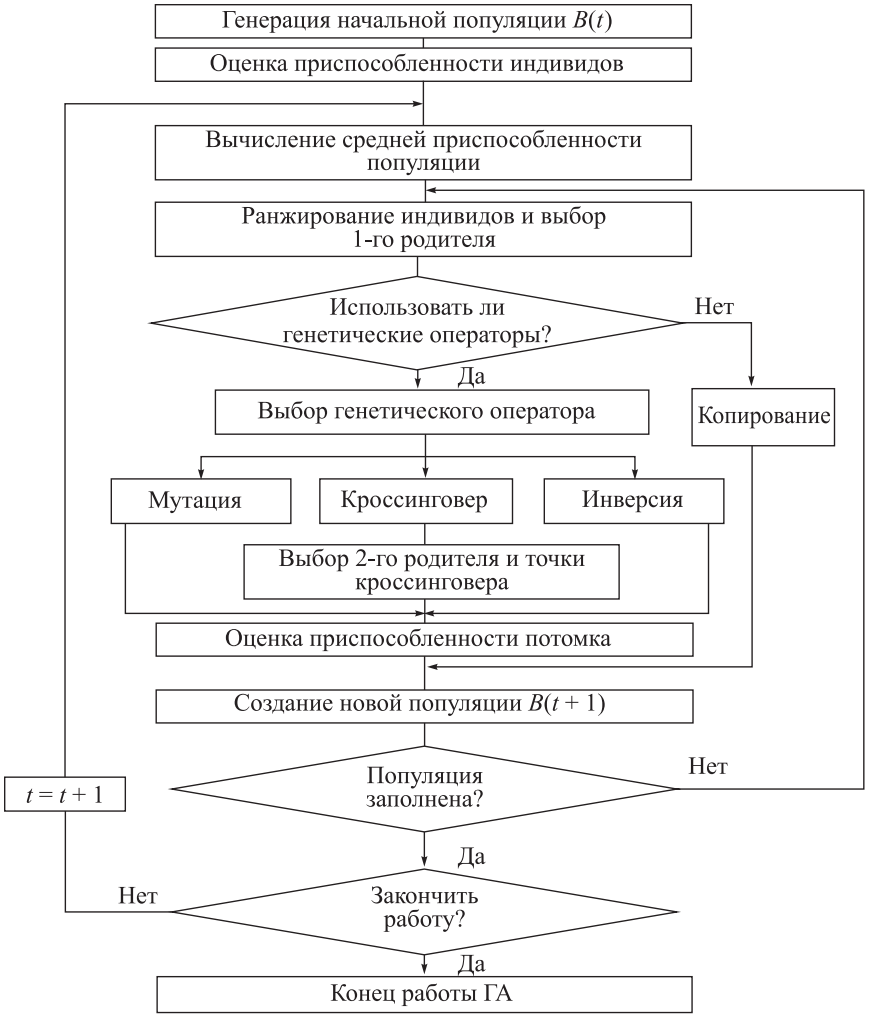


Рис. 5.25. Модифицированная схема работы генетического алгоритма

где $\boxed{\{7, 8, 9\}}$ — ген, выбранный для мутации. А затем к выбранному гену применяем мутацию:

7	8	9	ген до мутации

7	9	8	ген после мутации,

где || — точка мутации. После выполнения операции простой мутации получаем хромосому следующего вида:

{1, 2, 3}	{4, 5, 6}	{7, 9, 8}	{10, 11, 12}

После выполнения операции мутации последовательность ребер в гене 3 изменена, что может привести к разрыву цепи и получению нереального решения.

Покажем порядок выбора точки мутации с помощью чисел Фибоначчи. Пусть имеется векторная хромосома, представляющая некоторое число цепочек ребер. Для выполнения операции мутации случайным образом выбираем ген 3:

{1, 2, 3, 4, 5}	{6, 7, 8, 9, 10, 11}	{12, 13, 14, 15, 16, 17, 18, 19, 20}	{21, 22, 23, 24}

Затем к выбранному гену применяем оператор мутации, причем разряды для мутации выбираются в соответствии с числами Фибоначчи, т. е. 1, 2, 3, 5, 8, …, а обмен числовых значений между разрядами происходит по кругу со сдвигом вправо:

12	13	14	15	16	17	18	19	20

ген до мутации,

19	12	13	15	14	17	18	16	20

ген после мутации.

После выполнения операции мутации получаем хромосому следующего вида:

{1, 2, 3, 4, 5}	{6, 7, 8, 9, 10, 11}	{19, 12, 13, 15, 14, 17, 18, 16, 20}	{21, 22, 23, 24}

Рассматриваемый алгоритм имеет блочную структуру, где каждый блок является отдельным фрагментом, входные данные для которого предоставляет предыдущий блок (за исключением начального блока). Укрупненная структурная схема алгоритма представлена на рис. 5.26.

Опишем функции каждого блока. Первым блоком является ввод исходных данных. Он служит для преобразования заданного произвольным образом (матричным, графическим или иным образом) исходного графа к форме исходных данных, используемых алгоритмом.

Блок генерации циклов производит генерацию всех циклов заданной длины, выполняя это действие столько раз, сколько потребует от него блок анализа. т. е. он является своего рода поставщиком «сырья» для блока генетических операторов, в котором происходит процесс планаризации исследуемого графа. Блок укладки выполняет плоскую укладку графа либо его максимально планарной части.

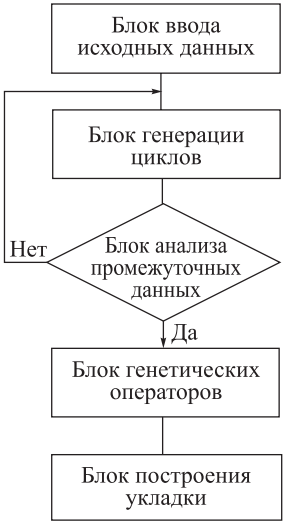


Рис. 5.26. Укрупненная структурная схема алгоритма планаризации

Исходный граф G может быть задан матрицей либо списком смежности. Матричное представление является избыточным. Для алгоритма определения планарности используется модифицированная форма списка смежности LS, где каждой вершине ставятся в соответствие номера ребер, инцидентных рассматриваемой вершине, например, запись $1:1,2$ означает, что ребро один инцидентно вершинам x_1 и x_2 и имеет номер 1, причем в список смежности некоторой вершины x_i не включаются номера вершин от x_1 до x_{i-1} включительно.

Например, для сгенерированного случайным образом графа G, изображенного на рис. 5.27, исходные данные для начала работы алгоритма запишутся так:

$$Lx_1 = \{1(1,2), 2(1,3), 3(1,5), 4(1,6), 5(1,8)\};$$
$$Lx_2 = \{6(2,3), 7(2,8)\};$$
$$Lx_3 = \{8(3,6), 9(3,8)\};$$
$$Lx_4 = \{10(4,6), 11(4,7), 12(4,8)\};$$
$$Lx_5 = \{13(5,6), 14(5,7), 15(5,8)\};$$
$$Lx_6 = \{16(6,7)\}.$$

Такая форма представления исходных данных не содержит избыточной информации и позволяет увеличить эффективность алгоритма.

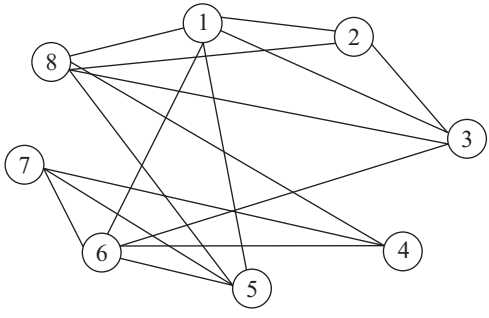


Рис. 5.27. Граф G, заданный случайным образом

Генерация циклов осуществляется на основе поиска в глубину. Число списков смежности будет меньше либо равно числу вершин графа, число элементов в списке Lx_i находится в интервале: $0 < |Lx_i| < \rho(x_i)$. Процесс генерации циклов k-й длины начинается со списка Lx_1, в котором выбирается вершина x_i, стоящая на первом месте, после чего выполняется переход к списку Lx_i. В случае, если на любом

из шагов обнаруживается невозможность дальнейшего продолжения (замыкания) цепи, алгоритм возвращается на один шаг назад и просматривает другие возможности для выбора пути. Таким образом формируется множество циклов длины k, представляющих собой цепочки вида $x_1 \dots x_i \dots x_k \dots x_1$. После того, как просмотрены все возможности и сформированы все циклы, использующие вершину x_1 в качестве начальной, список Lx_1 удаляется из рассмотрения, после чего процесс повторяется для следующего списка.

В процессе работы алгоритма генерируется множество всех возможных циклов длины k: $C_k = \{C_1, C_2, \dots, C_x\}$, где x — число циклов длины k. Каждый элемент множества C_k представляет собой цикл длины k, состоящий из номеров ребер, составляющих данный цикл. Так, например, для графа, изображенного на рис. 5.27, в процессе работы блока генерации циклов было получено следующее множество циклов длины три:

$$C_1 = (1-6-2);\quad C_2 = (1-7-5);\quad C_3 = (2-8-4);$$
$$C_4 = (2-9-5);\quad C_5 = (3-13-4);\quad C_6 = (3-15-5);$$
$$C_7 = (6-9-7);\quad C_8 = (10-16-11);\quad C_9 = (13-16-14).$$

На основе этих списков впоследствии может быть сформирована матрица циклов $B = \|b_{ij}\|_{c\times m}$, где c — число циклов, а m — число ребер графа, причем элемент матрицы $b_{ij} = 1$ в случае, если ребро m_j принадлежит циклу c_i ($m_j \in c_i$), и $b_{ij} = 0$ в противном случае. Пример матрицы циклов длины 3 показан на рис. 5.28.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C_1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
C_2	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
C_3	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
C_4	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
C_5	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0
C_6	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0
C_7	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0
C_8	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
C_9	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Рис. 5.28. Матрица циклов длины три

Полученные данные являются исходными для работы генетического алгоритма. Алгоритм анализа и принятия решений можно записать в следующем виде.

1. Подсчитать числа циклов $Nc_k = |C_k|$ и числа повторов Nm_j каждого ребра m_j, где $j = 1, \dots, M$.

2. Если выполняется условие $Nc_k \geqslant m - n + 2$ и $(\forall j \in M)(Nm_j \geqslant 2)$, то перейти к пункту 3, если нет, то вернуться к блоку генерации циклов и начать генерацию циклов длины $k + 1$.
3. Из множества циклов C_k выбрать первые $(m - n + 2)$ цикла и подсчитать общее число ребер Nm, принадлежащих этим циклам: $N_m = \sum\limits_{k=3}^{K} k \cdot |C_k|$, где k — длина цикла; $|C_k|$ — мощность множества циклов длины k; $K = k + s$ — длина самого длинного из сгенерированных циклов, где s — число шагов в блоке генерации циклов.
4. Полученное значение Nm сравнить с удвоенным числом ребер исходного графа G. Если $Nm > 2m$, то рассматриваемый граф является непланарным. Следовательно, необходимо принять решение. Если нужно выделить максимальную планарную часть графа и получить для нее плоскую укладку, то перейти к пункту 6, в противном случае — к 7.
5. Необходимо сформировать такую комбинацию циклов, чтобы выполнялось условие: $Nm = 2m$. Если полученное значение $Nm < 2m$, то необходимое условие достигается путем замены некоторого числа циклов меньшей длины (например, длины k) на циклы большей длины (циклы длины $k + 1$). Если циклы длины $k + 1$ отсутствуют, то нужно вернуться к блоку генерации циклов и начать генерацию циклов длины $k + 1$, в противном случае — перейти к пункту 6.
6. В случае, когда $Nc = 2m$, либо после преобразований, описанных в пункте 5, либо в случае, описанном в пункте 4, работу блока анализа завершить, полученные данные (множество циклов и предполагаемая схема комбинации) фиксировать и передать для работы генетического алгоритма.
7. Конец работы алгоритма

Для графа, представленного на рис. 5.27, процесс выбора решения будет таким:

1.1. Сформировано девять циклов длины три ($Nc_3 = 9$).

1.2. Необходимое число циклов: $p = m - n + 2 = 16 - 8 + 2 = 10$. Поскольку $Nc_3 < p$, то необходимо вернуться к блоку генерации циклов и начать генерацию циклов длины 4.

2.1. В результате работы блока генерации циклов сформировано 16 циклов длины четыре:

$C_{10} = (1{-}6{-}8{-}4)$; $C_{11} = (1{-}6{-}9{-}5)$; $C_{12} = (1{-}7{-}9{-}2)$;
$C_{13} = (1{-}7{-}15{-}3)$; $C_{14} = (2{-}6{-}7{-}5)$; $C_{15} = (2{-}8{-}13{-}3)$;
$C_{16} = (2{-}9{-}15{-}3)$; $C_{17} = (3{-}14{-}16{-}4)$; $C_{18} = (4{-}8{-}9{-}5)$;
$C_{19} = (4{-}10{-}12{-}5)$; $C_{20} = (4{-}13{-}15{-}5)$; $C_{21} = (8{-}10{-}12{-}9)$;
$C_{22} = (8{-}13{-}15{-}9)$; $C_{23} = 10{-}13{-}14{-}11)$; $C_{24} = (10{-}13{-}15{-}12)$;
$C_{25} = (11{-}14{-}15{-}12)$.

2.2. Общее число циклов стало равным двадцати пяти ($Nc = 9 + 16 = 25$), т. е. $Nc > p$. Число повторений каждого ребра $Nm_j > 2$, поэтому переходим к пункту 2.3.

2.3. Поскольку $p = 10$, то для первых десяти циклов подсчитываем число ребер: $Nc = 3 \times 9 + 4 \times 1 = 27 + 4 = 31$.

2.4. Число ребер Nm не больше удвоенного числа ребер графа $2m$ ($2m = 2 \times 16 = 32$), поэтому переходим к пункту 2.5.

2.5. Полученное значение Nm меньше удвоенного числа ребер графа ($2m = 2 \times 16 = 32 \Rightarrow Nc < 2m$). Формируем предполагаемую схему комбинации. Для этого необходимо заменить один из циклов длины три на цикл длины четыре.

2.6. После проведения замены условие $Nc = 2m$ выполнено и выясняется, что для решения поставленной задачи необходимо из имеющегося множества циклов выделить некоторый базис, включающий десять циклов, в том числе 8 циклов длины три и 2 — длины четыре.

6. Блок генетических операторов является основным блоком алгоритма планаризации, выполняющим исследование исходного графа на предмет определения планарности, а также нахождение вариантов его плоской укладки. Задача определения планарности и построения плоской укладки графа рассматривается как задача построения некоторого базиса (здания), удовлетворяющего установленным критериям (нормативам). Строительство базиса, так же как и строительство здания проводится в несколько этапов.

1 этап. «Строительство фундамента» — процесс формирования начального базового решения, которое должно стать основой для дальнейших преобразований.

2 этап. «Строительство каркаса» — это процесс дополнения имеющегося базового решения новыми комбинациями циклов до достижения максимально возможной степени заполнения. В зависимости от достигнутого на втором этапе результата (оптимальное или квазиоптимальное заполнение) процесс либо завершается, либо переходит к следующему этапу.

3 этап. «Индивидуальная достройка» — алгоритм пытается путем пошаговых изменений улучшить полученное ранее квазиоптимальное решение, т. е. найти глобальный оптимум. В случае, если это удается, процесс строительства завершается, в противном случае алгоритм переходит к следующему, четвертому и последнему этапу строительства.

4 этап. «Корректировка проекта». На этом этапе полученное квазиоптимальное решение, ввиду невозможности дальнейшего улучшения, запоминается, после чего в данное решение («проект») вносятся изменения и строительство возобновляется со второго этапа.

В качестве строительных методов на первых трех этапах используются генетические операторы селекции, кроссинговера и отбора, адаптированные к специфике поставленной задачи. Задача корректировки проекта выполняется с помощью операторов мутации и инверсии.

Сырьем для производства строительных блоков, используемых в процессе строительства, является множество циклов, сформированное блоком генерации циклов, причем строительные блоки могут быть размера два (число циклов в каждом из них) на первом и втором этапах работы генетического алгоритма либо размера один. Схема функционирования блока генетических операторов показана на рис. 5.29.

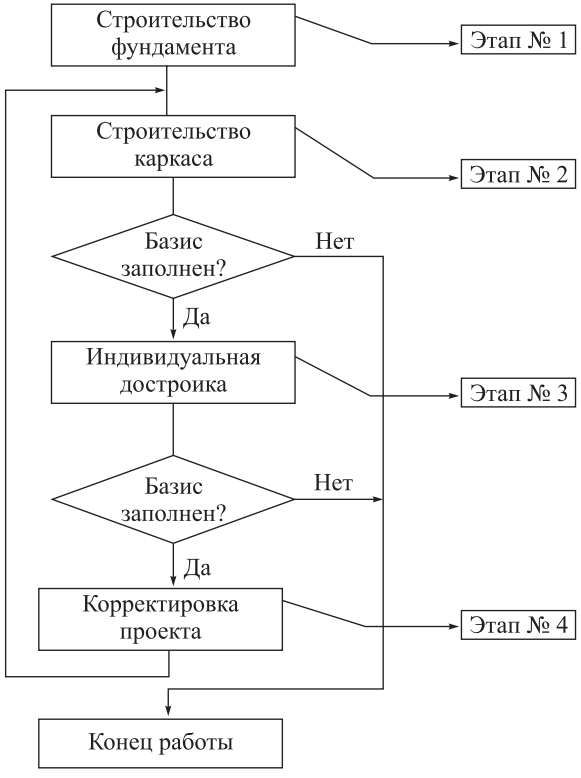


Рис. 5.29. Структурная схема генетического алгоритма

Исходными данными для рассматриваемого генетического алгоритма является множество циклов, из которого необходимо некоторым образом выделить базис циклов B_p, где $p = m - n + 2$. Иначе говоря, для представления базиса в виде хромосомы необходимо, чтобы в ней было не менее p разрядов (генов), каждый из которых соответствует номеру цикла. Таким образом, под хромосомой понимается комбинация из p циклов, являющаяся одновременно вариантом плоской укладки исходного графа.

В алгоритме используются специального вида хромосомы, с числом генов равным $(p + 3)$, причем каждая хромосома в популяции состоит как бы из двух частей. Левая часть хромосомы имеет p разрядов,

представляющих собой комбинацию задействованных в данном варианте решения циклов, а правая часть имеет три разряда, несущих информацию о том, какие ребра и сколько раз содержатся в выбранной комбинации (0, 1 или 2 раза соответственно для разрядов $p+1$, $p+2$ и $p+3$). Кроме того, последние три разряда используются при количественной оценке качества полученного решения. Для этого вводится в рассмотрение функция $\delta(m)$, которая стремится к минимуму для генов $p+1$ и $p+2$ ($f(H) \to 0$) и стремится к максимуму для гена $p+3$ ($f(H) \to m$):

$$\delta(m) = (2 \times Nm_{p+3} + Nm_p + 2)/2m,$$

где Nm_p+3 — число элементов в p+3 разряде хромосомы, а Nm_p+2 — соответственно число элементов в $p+2$ разряде хромосомы.

Как следует из формулы, значения, которые может принимать функция оценки $\delta(m)$, находятся в интервале $[0, 1]$, причем качество решения будет тем выше, чем больше число ребер и прежде всего ребер, встречающихся дважды, задействовано в оцениваемом решении. Левая часть хромосомы, в свою очередь, может содержать как значащую, так и незначащую части. Разряды значащей части заполняются номерами циклов, а незначащей — нулями. При этом чем выше качество решения, тем меньше длина незначащей части ($L_0 \to 0$). Для оптимального решения L_0 будет равна нулю. Таким образом, целевая функция хромосомы полученного решения $f(H)$ является аддитивной функцией двух переменных: $f(H) = (\delta(m);\ L_0)$. Причем $f(H) \to \max$, когда $\delta(m) \to 2m$ и $L_0 \to 0$.

Методику расчета целевой функции хромосомы покажем на следующем примере. Пусть существует некая произвольная комбинация циклов C_1, C_2 и C_3, закодированная хромосомой следующего вида:

1	**2**	**3**	0	0	0	0	0	0	0	{**3,9,10,11,12,13,14,15,16**}	{**4,5,6,7,8**}	{**1,2**}

Тогда целевая функция данной хромосомы равна $f(H) = (0,2613;7)$: $L_0 = 7$; $\delta(m) = (2 \times 2 + 5)/(2 \times 16) = 9/32 = 0{,}2813$. Решение, имеющее подобное значение целевой функции, соответствует промежуточному решению, поскольку значение целевой функции базового решения не может быть менее 0,5.

Для алгоритма используются два варианта стратегии создания начального базового решения, избираемые в зависимости от характера исходных данных и решения ЛПР.

Первый вариант заключается в том, что в имеющемся множестве циклов C выбираем ребра, повторяющиеся точно два раза. Если это не удается сделать, то область поиска сужается и просматриваются только первые p циклов. Циклы, которым принадлежат ребра, суммируются, образуя определенную комбинацию. Эта комбинация проверяется на предмет образования нереальных решений, т. е. наличия ребер,

повторяющихся более двух раз. Если такие ребра обнаруживаются, то из комбинации удаляется минимально возможное число циклов, содержащих «лишние» ребра, а полученная комбинация преобразуется в хромосому. Таким образом, получается начальное базовое решение.

Второй вариант стратегии создания начального множества решений реализуется в случаях, когда не удалось создать начальное базовое решение (т. е. выделить начальный базис циклов, содержащих точно две единицы) либо когда процесс улучшения базового решения зашел в тупик (локальный оптимум) и в течение нескольких поколений целевая функция базового решения не улучшается. Тогда начальное базовое решение формируется случайным образом, путем последовательного объединения циклов, имеющих хотя бы одну одинаковую вершину.

Рассмотрим подробнее первый вариант стратегии создания начального множества решений на примере матрицы циклов графа G, сгенерированного случайным образом (рис. 5.27).

В результате работы алгоритма сформировано множество циклов C_k длины 3 и 4, где $Nc = 25$. В нем ищем ребра, повторяющиеся точно два раза. Так как таких ребер нет $(\forall j \in M)\,(Nm_j > 2)$, то сужаем область поиска и просматриваем первые p элементов множества C. В результате поиска выясняется, что таких ребер четыре. Это ребра номер 3, 7, 13, 16. Теперь суммируем все циклы, включающие указанные ребра. Получаем комбинацию циклов: $\{C_2, C_5, C_6, C_7, C_8, C_9\}$, а затем проверяем, не создает ли полученная комбинация нереальных решений. Поскольку ни в одном столбце сумма единиц не превышает два, данная комбинация является начальным базовым решением и будет закодирована следующей хромосомой:

1н	**2**	**5**	**6**	**7**	**8**	**9**	0	0	0	0	{**2,8,12**}	{**1,4,6,9,10,11,14,15**}	{**3,5,7,13,16**}

Целевая функция хромосомы 1н равна $\delta(m) = (2 \times 5 + 8)/32 = 18/32 = 0{,}5625$; $L_0 = 4$; $f(1\text{н}) = (0{,}5625; 4)$. На данном этапе продолжается процесс заполнения сформированного ранее начального базового решения с помощью генетических операторов селекции, кроссинговера и отбора. Схема работы генетического алгоритма на этом этапе выглядит следующим образом (рис. 5.30):

Шаг «А». Имеется хромосома, представляющая начальное базовое решение. Теперь необходимо сформировать множество промежуточных решений, из числа которых будет выбран второй родитель для участия в операции скрещивания. Множество промежуточных решений формируется путем попарного объединения циклов, причем объединение производится таким образом, чтобы образующиеся в результате парные ребра были идентичны ребрам, присутствующим в разрядах $p + 1$, $p + 2$ начального базового решения. Кроме того, номера циклов множества промежуточных решений не должны совпадать с номерами циклов, участвующих в базовом решении. Размер популяции промежуточных решений зависит от числа ребер в разрядах $p + 1$, $p + 2$ базового

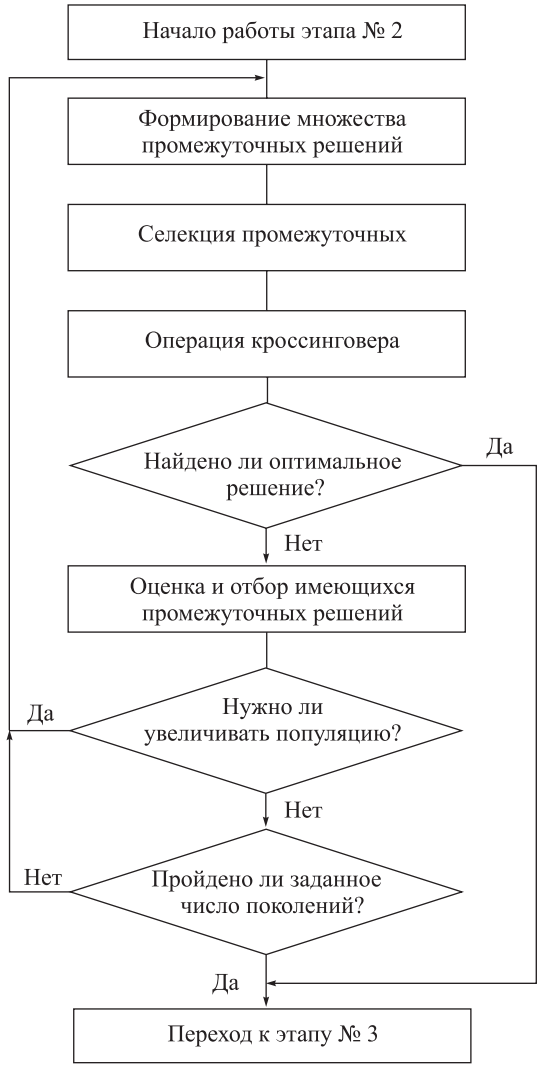


Рис. 5.30. Структурная схема генетического алгоритма (этап № 2)

решения и от длины хромосомы L (т. е. размерности исследуемого графа). Для рассматриваемого примера множество промежуточных решений формируется случайным образом, путем попарного объединения циклов, включающих ребра 2, 8, 12.

Размер начальной популяции примем равным трем (по одной хромосоме для каждого ребра):

2н	**1**	**3**	0	0	0	0	0	0	0	0	{**3,5,7,9,10,11,12,13,14,15,16**}	{**1,4,6,8**}	{**2**}

3н	**15**	**18**	0	0	0	0	0	0	0	0	{**1,6,7,10,11,12,14,15**}	{**2,3,4,5,9,13,16**}	{**8**}
4н	**19**	**21**	0	0	0	0	0	0	0	0	{**1,2,3,6,7,13,14,15,16**}	{**4,5,8,9,11**}	{**10,12**}

После того, как множество промежуточных решений сформировано, алгоритм переходит на следующий шаг.

Шаг «В». Здесь производится селекция множества промежуточных решений с целью выбора хромосомы. Для оценки каждого решения были разработаны следующие правила:

1. В генах $p+2$ и $p+3$ претендента, по возможности, не должно быть ребер, совпадающих с ребрами из разряда $p+3$ первого родителя.
2. В генах $p+1$ претендента и $p+3$ первого родителя желательно иметь максимальное количество совпадающих ребер.
3. В гене $p+2$ претендента должно быть максимальное число ребер, идентичных ребрам из разряда $p+2$ хромосомы первого родителя.

На основе правил создан комплексный параметр — функция пригодности промежуточных решений относительно базового решения $f'(H)$. Данный параметр является аддитивной функцией двух переменных $f'(H) = (\delta' m; N' c)$, где $\delta' m$ — относительное увеличение числа ребер; $N' c$ — относительное увеличение числа циклов. Количественное значение функции пригодности рассчитывается следующим образом:

а) определяется число совпадений Nd_1 ребер из разряда $p+3$ оцениваемого промежуточного решения с ребрами из разряда $p+3$ первого родителя. Затем подсчитывается число совпадений Nd_2 в разрядах $p+3$ промежуточного решения и $p+2$ первого родителя, а также число совпадений Nd_3 в разрядах $p+2$ промежуточного решения и $p+3$ родителя. Сумма этих чисел характеризует возможное число ребер, которые необходимо будет удалить из базового решения после выполнения кроссинговера. Следовательно, в оптимальных промежуточных решениях значение суммы будет стремиться к нулю;

б) находится число совпадений Ns_1 ребер из разряда $p+3$ оцениваемого промежуточного решения с ребрами из разряда $p+1$ первого родителя. Затем подсчитывается число совпадений Ns_2 в разрядах $p+2$ промежуточного решения и $p+1$ первого родителя и, наконец, подсчитывается число совпадений Ns_3 в разрядах $p+2$ промежуточного решения и $p+2$ родителя. В этом случае определяется число ребер, которые добавляются в базовое решение в результате операции кроссинговера, т. е. качество промежуточного решения тем выше, чем больше значение этой суммы;

в) подсчитывается значение $\delta' m$ по формуле: $\delta' m = 1 - [(2 \times Nd_1 + Nd_2 + Nd_3)/(2 \times Ns_1 + Ns_2 + Ns_3)]$. Полученное значение зависит от соотношения возможного числа удаляемых и добавляемых ребер и стремится к единице для наилучших промежуточных решений. В случае получения отрицательного значения функции пригодности делается вывод о том, что данное решение непригодно для участия в дальнейших операциях;

г) сравниваются ребра в разрядах $p+2$ и $p+3$ промежуточного решения с ребрами в разряде $p+3$ базового решения и, при наличии совпадений, выявляется минимальное число циклов $Nc-$, которые после выполнения оператора кроссинговера будут удалены из базовой хромосомы для устранения нереальных решений, поэтому в оптимальном промежуточном решении значение $N'c$ будет стремиться к нулю;

д) вычисляется значение $N'c$ по формуле: $N'c = 1 - (Nc - /Nc+)$.

е) подсчитывается значение функции пригодности хромосомы относительно текущего базового решения $f'(H)$.

После этого из множества промежуточных решений выбирается элемент, имеющий наилучшую функцию пригодности относительно первого родителя и назначается в пару к базовому решению для выполнения оператора кроссинговера.

Для рассматриваемого примера подсчитаем значения функций пригодности относительно базового решения (хромосома 1н):

$$\begin{aligned}
\delta' m(2\text{н}) =& 1 - [(0+0)/(2\times 1+4)] = 1;\\
& N'c(2\text{н}) = 1 - (0/2) = 1; \quad f'(H) = (1;1).\\
\delta' m(3\text{н}) =& 1 - [(0+4)/(2\times 1+1+2)] = 1 - (4/5) = 1 - 0,8 = 0,2;\\
& N'c(3\text{н}) = 1 - (2/2) = 1 - 0 = 0; \quad f'(H) = (0,2;0);\\
\delta' m4\text{н}) =& 1 - [(2\times 1+1)/(2\times 1+1+3)] = 1 - (3/6) = 1 - 0,5 = 0,5;\\
& N'c(4\text{н}) = 1 - (2/2) = 0; \quad f'(H) = (0,5;0).
\end{aligned}$$

После ранжирования полученных функций пригодности промежуточных решений в качестве второго родителя будет выбрана хромосома 2н, как имеющая наивысшую оценку пригодности.

Шаг «С». На этом шаге выполняется оператор кроссинговера двух родительских особей. Здесь используется двухточечный суммирующий оператор кроссинговера. Точки скрещивания в операторе выбираются следующим образом:

1. В первом родителе первая точка скрещивания t_1 находится за последним значащим (ненулевым) элементом h_i хромосомы H, а позиция второй точки t_2 выбирается после элемента $h_i + k$, где k — число значащих элементов во второй родительской хромосоме.

2. Во втором родителе первая точка находится перед первым элементом хромосомы, а вторая точка определяется так же, как в случае с первым родителем.

В результате выполнения оператора кроссинговера происходит объединение значащих частей родительских хромосом. В случае, если в функции пригодности второго родителя $\delta' m(H) \neq 0$, то из хромосомы потомка удаляются ребра (или ребро), которые повторяются в данной комбинации более двух раз. Параллельно с удалением ребер удаляются Nc циклов, которым принадлежит данное ребро. Данные циклы (цикл) вместе с принадлежащими им ребрами заносятся в хромосому второго родителя. При этом вновь появившиеся в хромосоме циклы не могут

быть удалены. В примере точки операторы кроссинговера для хромосомы 1н: $t_1 = 6$, $t_2 = 8$, и для хромосомы 2н: $t_1 = 0$, $t_2 = 2$.

1н	**2**	**5**	**6**	**7**	**8**	**9**	0	0	0	0	{**2,8,12**}	{**1,4,6,9,10,11,14,15**}	{**3,5,7,13,16**}
2н	**1**	**3**	0	0	0	0	0	0	0	0	{**3,5,7,9,10,11,12,13,14,15,16**}	{**1,4,6,8**}	{**2**}

После выполнения оператора кроссинговера получается новое базовое решение:

1п	**2**	**5**	**6**	**7**	**8**	**9**	**1**	**3**	0	0	{**12**}	{**8,9,10,11,14,15**}	{**1,2,3,4,5,6,7,13,16**}

Шаг «D». Здесь популяция промежуточных решений переупорядочивается. Для этого производится пересчет функций пригодности каждого элемента популяции относительно нового базового решения. В случае, если после произведенного пересчета функции пригодности всех элементов имеют низкие оценки ($f'(H) = (\delta' m(H) < 25\,\%$ и $N'c(H) < 0)$), то необходимо либо расширить размер популяции промежуточных значений, либо перейти к следующему этапу строительства. Решение принимается ЛПР на основании оценки размера популяции и числа пройденных поколений. Расширение размера популяции производится за счет добавления промежуточных решений, включающих пары ребер, идентичных ребрам из разряда $p + 3$ базовой хромосомы. Если получено базовое решение, в котором число ребер в разряде $p + 1$ равно нулю ($Nm_{p+1} = 0$), а в разряде $p + 2$ $Nm_{p+2} \neq 0$, то алгоритм также переходит к следующему этапу.

Поскольку размер популяции промежуточных решений в рассматриваемом примере изначально мал и число пройденных поколений ($N_G = 1$) невелико, увеличиваем популяцию промежуточных решений за счет добавления хромосом, содержащих ребро 12, так как в разряде $p + 1$ базового решения осталось только одно это ребро:

5н	**19**	**24**	0	0	0	0	0	0	0	0	{**1,2,3,6,7,8,9,11,14,16**}	{**4,5,13,15**}	{**10,12**}
6н	**21**	**25**	0	0	0	0	0	0	0	0	{**1,2,3,4,5,6,7,13,16**}	{**8,9,10,11,14,15**}	{**12**}
7н	**19**	**25**	0	0	0	0	0	0	0	0	{**1,2,3,6,7,8,9,13,16**}	{**4,5,10,11,14,15**}	{**12**}
8н	**21**	**24**	0	0	0	0	0	0	0	0	{**1,2,3,4,5,6,7,11,14,16**}	{**8,9,13,15**}	{**10,12**}

Подсчитаем функции пригодности новых промежуточных решений:

$$\begin{array}{lll} \delta' m(5\text{н}) = 0{,}2; & N'c(3\text{н}) = 0; & f'(H) = (0{,}2; 0); \\ \delta' m(6\text{н}) = 1; & N'c(6\text{н}) = 1; & \\ \delta' m(7\text{н}) = 0{,}6667; & N'c(7\text{н}) = 0; & \\ \delta' m(8\text{н}) = 0{,}6; & N'c(8\text{н})0. & \end{array}$$

Лучшую оценку пригодности имеет хромосома 6н, которая будет выбрана в качестве второго родителя для операторы кроссинговера. Теперь возвращаемся к этапу «C» и выполняем операторы кроссинговера для хромосом 1п и 6н:

1п	**2**	**5**	**6**	**7**	**8**	**9**	**1**	**3**	0	0	{**12**}	{**8,9,10,11,14,15**}	{**1,2,3,4,5,6,7,13,16**}

6н	**21**	**25**	0	0	0	0	0	0	0	0	{**1,2,3,4,5,6,7,13,16**}	{**8,9,10,11,14,15**}	{**12**}

2п	**2**	**5**	**6**	**7**	**8**	**9**	**1**	**3**	**21**	**25**	{**0**}	{**0**}	{**1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16**}

В полученной хромосоме 2п все разряды с 1-го по p-й (т. е. с 1-го по 10-й) являются значащими, а разряды $p+1$ и $p+2$ — пустые. Следовательно, условие планарности графа выполнено. Таким образом, в результате выполнения оператора кроссинговера построено оптимальное базовое решение (хромосома 2п).

После декодирования хромосомы будет сформирован необходимый базис циклов B_p (рис. 5.31). Следовательно, данный граф может быть уложен на плоскости без пересечений.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
C_1	**1**	**1**	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
C_2	**1**	0	0	0	**1**	0	**1**	0	0	0	0	0	0	0	0	0
C_3	0	**1**	0	**1**	0	0	0	**1**	0	0	0	0	0	0	0	0
C_5	0	0	**1**	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0
C_6	0	0	**1**	0	**1**	0	0	0	0	0	0	0	0	0	**1**	0
C_7	0	0	0	0	0	**1**	**1**	0	**1**	0	0	0	0	0	0	0
C_8	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	**1**
C_9	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	**1**
C_{21}	0	0	0	0	0	0	0	**1**	**1**	**1**	0	**1**	0	0	0	0
C_{25}	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	**1**	**1**	0

Рис. 5.31. Базис циклов B_P

Задача алгоритма на этапе 3 состоит в том, чтобы при имеющемся квазиоптимальном решении поиск оптимального решения вести, не ухудшая качества имеющегося. Для этого популяция промежуточных решений расщепляется таким образом, чтобы каждая хромосома новой популяции содержала в себе только один цикл. После чего выполняются действия, идентичные шагам «B»–«D» этапа 2. При построении оптимального решения алгоритм завершает работу, в противном случае

переход к этапу 4. Переход к этапу 4 осуществляется, если в ходе решения задачи получено квазиоптимальное решение, не улучшающееся на протяжении значительного числа поколений (попадание в локальный оптимум).

В алгоритме за основу был взят стандартный оператор — мутация обмена. Для проведения оператора мутации случайным образом выбираются два гена, один из которых является значащим элементом, а второй — незначащим. В ходе выполнения оператора мутации элементы меняются местами, после чего значащий элемент обнуляется, все принадлежащие данному циклу ребра удаляются из генов $p+2$ и $p+3$ и, при необходимости, оставшиеся значащие элементы сдвигаются вправо, вытесняя нулевой элемент из значащей части. Например, для хромосомы 1п оператор мутации выполняется следующим образом. В хромосоме случайным образом выбираем два гена, один из которых значащий, а второй нулевой. Пусть это будут гены 3 и 10.

1п	**2**	**5**	**6**	**7**	**8**	**9**	**1**	**3**	0	0	{**12**}	{**8,9,10,11,14,15**}	{**1,2,3,4,5,6,7,13,16**}

После этого производим обмен:

1п	**2**	**5**	0	**7**	**8**	**9**	**1**	**3**	0	**6**	{**12**}	{**8,9,10,11,14,15**}	{**1,2,3,4,5,6,7,13,16**}

Из генов $p+2$ и $p+3$ удаляем ребра принадлежащие циклу 6.

1пм	**2**	**5**	0	**7**	**8**	**9**	**1**	**3**	0	0	{**12,15**}	{**3,5,8,9,10,11,14**}	{**1,2,4,6,7,13,16**}

Сдвигаем значащие элементы влево, после чего оператором мутации заканчивает свою работу:

1пм	**2**	**5**	**7**	**8**	**9**	**1**	**3**	0	0	0	{**12,15**}	{**3,5,8,9,10,11,14**}	{**1,2,4,6,7,13,16**}

Схема проведения оператора инверсии аналогична оператору мутации с той лишь разницей, что для обмена выбирается не один ген хромосомы, а несколько. Число обмениваемых генов задается случайным образом, однако оно не должно быть больше числа незначащих разрядов инвертируемой хромосомы. Вероятности проведения операций мутации $\Pr(\text{ОМ})$ и инверсии $\Pr(\text{ОИ})$ выбираются случайным образом на интервале $[0,1]$, причем их суммарная вероятность $\Pr(\text{SUM}) = \Pr(\text{ОМ}) + \Pr(\text{ОИ}) = 1$.

Для построения плоской укладки по сформированному базису циклов B_p (рис. 5.31) используем следующий принцип. Согласно ему плоская укладка строится следующим образом: в базисе p последовательно выбираются цикл за циклом, причем на каждом шаге разрастание происходит за счет включения новых ребер во внешний цикл. Таким образом, на каждом шаге внешний цикл либо увеличивается, если

в новом цикле число новых, незадействованных ребер, больше числа повторяющихся, либо уменьшается — в противном случае. При этом повторяющиеся ребра автоматически выпадают из внешнего цикла и оказываются внутри.

На каждом $(i+1)$-м шаге находим разность множеств $C'_{\text{вн}_i} = C_{\text{вн}_i} \backslash C_{i+1}$ и $C'_{i+1} = C_{i+1} \backslash C_{\text{вн}_i}$, где $C_{\text{вн}}$ — внешний цикл графа, после чего выполняем суммирование $C_{\text{вн}_{i+1}} = C'_{\text{вн}_i} + C_{i+1}$'. Работа алгоритма плоской укладки заканчивается после того, как все ребра графа будут пройдены как минимум один раз. Покажем работу алгоритма плоской укладки для графа G (рис. 5.27). Выбираем первый цикл в подматрице p. Это цикл C_1. На первом шаге цикл C_1 совпадает с внешним циклом $C_{\text{вн}_1}$; $C_{\text{ин}_1} = \{1(1,2) - 6(2,3) - 2(3,1)\}$. Затем выбираем следующий цикл в базисе p. Это цикл C_2. Находим разность множеств: $C'_{\text{вн}_1} = C_{\text{вн}_1} \backslash C_2 = \{6(2,3), 2(3,1)\}$ и $C'_2 = C_2 \backslash C_{\text{вн}_1} = \{5(1,8), 7(8,2)\}$. После суммирования $C'_{\text{вн}_1}$ и C_2' получим: $C_{\text{вн}_2} = \{5(1,8) - 7(8,2) - 6(2,3) - 2(3,1)\}$. Далее процесс продолжается аналогичным образом:

$$
\begin{aligned}
C_{\text{вн}_3} &= \{5(1,8) - 7(8,2) - 6(2,3) - 8(3,6) - 4(6,1)\};\\
C_{\text{вн}_4} &= \{5(1,8) - 7(8,2) - 6(2,3) - 8(3,6) - 13(6,5) - 3(5,1)\};\\
C_{\text{вн}_5} &= \{7(8,2) - 6(2,3) - 8(3,6) - 13(6,5) - 15(5,8)\};\\
C_{\text{вн}_6} &= \{9(8,3) - 8(3,6) - 13(6,5) - 15(5,8)\}.
\end{aligned}
$$

На следующем шаге алгоритма должен быть выбран цикл C_8, однако в нем нет ни одного ребра, принадлежащего внешнему циклу. Поэтому временно пропустим его и перейдем к циклу C_9. $C_{\text{вн}_7} = \{9(8,3) - 8(3,6) - 16(6,7) - 14(7,5) - 15(5,8)\}$. Теперь во внешнем цикле появилось ребро 166,7 и возвращаемся к циклу C_8:

$$
\begin{aligned}
C_{\text{вн}_8} &= \{9(8,3) - 8(3,6) - 10(6,4) - 11(4,7) - 14(7,5) - 15(5,8)\};\\
C_{\text{вн}_9} &= \{12(8,4) - 11(4,7) - 14(7,5) - 15(5,8)\}.
\end{aligned}
$$

Алгоритм плоской укладки завершил работу, так как все ребра графа пройдены. В результате получена плоская укладка исходного графа G (рис. 5.32).

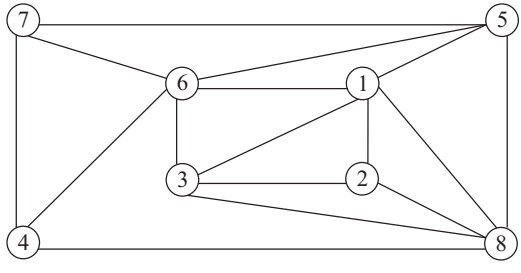


Рис. 5.32. Плоская укладка графа G

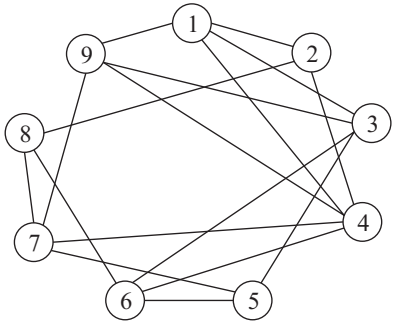


Рис. 5.33. Граф G', заданный случайным образом

Рассмотрим теперь применение описанного алгоритма для решения задачи минимизации пересечений в непланарном графе на примере сгенерированного случайным образом графа G' (рис. 5.33). Списки смежности данного графа будут выглядеть следующим образом:

$$Lx_1 = \{1(1,2), 2(1,3), 3(1,4), 4(1,9)\};$$
$$Lx_2 = \{5(2,4), 6(2,8)\};$$
$$Lx_3 = \{7(3,5), 8(3,6), 9(3,3)\};$$
$$Lx_4 = \{10(4,6), 11(4,7), 12(4,9)\};$$
$$Lx_5 = \{13(5,6), 14(5,7)\};$$
$$Lx_6 = \{15(6,8)\};$$
$$Lx_7 = \{16(7,8), 17(7,9)\}.$$

После работы блока генерации циклов будет сформировано следующее множество циклов длины три:

$$C_1 = (1-5-3);$$
$$C_2 = (2-9-4);$$
$$C_3 = (3-12-4);$$
$$C_4 = (7-13-8);$$
$$C_5 = (11-17-12).$$

Поскольку $Nc < p$, то продолжаем генерацию циклов длины четыре:

$$C_6 = (1-5-12-4);$$
$$C_7 = (2-8-10-3);$$
$$C_8 = (2-9-12-3);$$

$$C_9 = (3 - 11 - 17 - 4);$$
$$C_{10} = (5 - 10 - 15 - 6);$$
$$C_{11} = (5 - 11 - 16 - 6);$$
$$C_{12} = (7 - 14 - 17 - 9);$$
$$C_{13} = (8 - 10 - 12 - 9);$$
$$C_{14} = (10 - 13 - 14 - 11);$$
$$C_{15} = (10 - 15 - 16 - 11);$$
$$C_{16} = (13 - 15 - 16 - 14).$$

В результате общее число сгенерированных циклов Nc равно 16. Переходим к блоку анализа (п. 2) и сравниваем: $Nc > p$; $Nm_j > 2$. Условие выполнено, переходим к пункту 3. Выбираем первые 10 циклов и подсчитываем число ребер: $5 \times 3 + 5 \times 4 = 35$; $2 \times m = 2 \times 17 = 34$; $35 > 34$. Следовательно, граф непланарный. Для минимизации числа пересечений ребер данного графа переходим к блоку генетических операторов.

Создание начального базового решения выполним по второму варианту стратегии. Для этого последовательно просматриваем имеющиеся списки и выбираем попарно циклы, имеющие как минимум одно общее ребро, а затем суммируем их. В результате получится следующее множество начальных решений:

1н	**1**	**6**	0	0	0	0	0	0	0	{**2,6–11,13–17**}	{**3,4,12**}	{**1,5**}
2н	**2**	**7**	0	0	0	0	0	0	0	{**1,5,6,7,11–17**}	{**3,4,8,9,10**}	{**2**}
3н	**4**	**12**	0	0	0	0	0	0	0	{**1–6,10–12,15,16**}	{**8,9,13,14,17**}	{**7**}
4н	**5**	**14**	0	0	0	0	0	0	0	{**1–9,15,16**}	{**10,12,13,14,17**}	{**11**}

Просуммировав их, получим начальное базовое решение:

5н	**1**	**6**	**2**	**7**	**4**	**12**	**5**	**14**	0	0	{**6,15,16**}	{**0**}	{**1–5,7–14,17**}

Целевая функция полученного базового начального решения равна

$$L_0 = 2; \quad \delta(m) = (2 \times 14 + 0)/(2 \times 17) = 28/38 = 0{,}8235;$$
$$f(H) = (0{,}8235; 2).$$

Теперь необходимо сформировать популяцию промежуточных решений. Поскольку в начальном базовом решении отсутствуют 6 ребер, примем размер популяции равным 6:

6н | **10** | **11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | {**1–4,7–9,12,13,14,17**} | {**10,11,15,16**} | {**5,6**}

7н | **15** | **16** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | {**1–9,17**} | {**10,11,13,14**} | {**15,16**}

8н | **10** | **15** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | {**1–4,7–9,12–14,17**} | {**5,6,11,16**} | {**10,15**}

9н | **11** | **16** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | {**1–4,7–10,12,17**} | {**5,6,11,13,14,15**} | {**16**}

10н | **10** | **16** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | {**1–4,7–9,11,12,17**} | {**5,6,10,13,14,16**} | {**15**}

11н | **11** | **15** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | {**1–4,7–9,12–14,17**} | {**5,6,10,15**} | {**11,16**}

Теперь вычисляем функции пригодности полученных хромосом:

$$\begin{array}{lll}
\delta' m(6\text{н}) = 0; & N' c(6\text{н}) = -0{,}5; & f'(6\text{н}) = (0; -0{,}5);\\
\delta' m(7\text{н}) = 0; & N' c(7\text{н}) = 0{,}5; & f'(7\text{н}) = (0; 0{,}5);\\
\delta' m(8\text{н}) = 0; & N' c(8\text{н}) = -0{,}5; & f'(8\text{н}) = (0; -0{,}5);\\
\delta' m(9\text{н}) = 0; & N' c(9\text{н}) = 0; & f'(9\text{н}) = (0; 0);\\
\delta' m(10\text{н}) = 0; & N' c(10\text{н}) = 0; & f'(10\text{н}) = (0; 0);\\
\delta' m(11\text{н}) = 0; & N' c(11\text{н}) = -0{,}5; & f'(11\text{н}) = (0; -0{,}5).
\end{array}$$

Согласно полученным функциям пригодности, хромосомы 6, 8, 11 удаляются из популяции. Из трех оставшихся хромосом наибольшее значение функции пригодности имеет хромосома 7 ($f'(7\text{н}) = (0; 0{,}5)$), которая и будет отобрана для проведения оператора кроссинговера:

5н | **1** | **6** | **2** | **7** | **4** | **12** | **5** | **14** | 0 | 0 | {**6,15,16**} | {**0**} | {**1–5,7–14,17**}

7н | **15** | **16** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | {**1–9,17**} | {**10,11,13,14**} | {**15,16**}

1п | **1** | **6** | **2** | **7** | **4** | **12** | **5** | **14** | **15** | **16** | {**6**} | {**0**} | {**1–5,7–10,11,13,14–17**}

В полученной хромосоме (1п) есть четыре ребра (10, 11, 13, 14), повторяющиеся трижды. Для того, чтобы избежать появления нереального решения, достаточно удалить один цикл 14. Тогда базовое решение примет следующий вид:

1п | **1** | **6** | **2** | **7** | **4** | **12** | **5** | **15** | **16** | 0 | {**6**} | {**0**} | {**1–5,7–17**}

Определим целевую функцию полученного базового решения:

$L_0 = 1$; $\delta(m) = (2 \bullet 16)/(2 \bullet 17) = 28/38 = 0{,}9412$; $f(H) = (0{,}9412; 1)$.

Как видно, значение целевой функции базового решения улучшилось. Полученное значение $f(H)$ позволяет считать, что найдено некото-

рое квазиоптимальное решение. Однако базис циклов сформирован не полностью. Продолжим процесс исследования. Для этого применяется оператор мутации. Для проведения оператора мутации случайным образом выбираем разряд 7 в хромосоме базового решения и обнуляем его:

1п	**1**	**6**	**2**	**7**	**4**	**12**	0	**15**	**16**	0	{**6**}	{**0**}	{**1–5,7–17**}

Затем из правой части хромосомы удаляем ребра, соответствующие циклу 5:

2п	**1**	**6**	**2**	**7**	**4**	**12**	**15**	**16**	0	0	{**6**}	{**11,12,17**}	{**1–5,7–10,13–16**}

Для полученной хромосомы (2п), используя стратегию доработки базового решения, сформируем множество промежуточных решений. Это множество решений, состоящих из одного цикла:

$1н_2$	**3**	0	0	0	0	0	0	0	0	0	{**1,2,5–11,13–17**}	{**3,4,12**}	{**0**}
$2н_2$	**8**	0	0	0	0	0	0	0	0	0	{**1,5–8,10,11,13–17**}	{**2,3,9,12**}	{**0**}
$3н_2$	**9**	0	0	0	0	0	0	0	0	0	{**1,2,5–10,12–16**}	{**3,4,11,17**}	{**0**}
$4н_2$	**10**	0	0	0	0	0	0	0	0	0	{**1–4,7–9,11–14,16,17**}	{**5,6,10,15**}	{**0**}
$5н_2$	**11**	0	0	0	0	0	0	0	0	0	{**1–4,7–10,12–15,17**}	{**5,6,11,16**}	{**0**}
$6н_2$	**13**	0	0	0	0	0	0	0	0	0	{**1–7,11,13–17**}	{**8,9,10,12**}	{**0**}

Следовательно, построенное базовое решение является оптимальным для данного графа и улучшено быть не может.

Построим теперь максимально планарную укладку данного графа. В результате работы генетического алгоритма сформирован следующий базис циклов:

$$B_p = \{C_1, C_6, C_2, C_7, C_4, C_{12}, C_5, C_{15}, C_{16}\}.$$

Выбираем первый цикл базиса C_1.

$$C_{\text{вн}_1} = \{1(1,2) - 5(2,4) - 3(4,1)\};$$
$$C_{\text{вн}_2} = \{4(1,9) - 12(9,4) - 3(4,1)\};$$
$$C_{\text{вн}_3} = \{2(1,3) - 9(3,9) - 12(9,4) - 3(4,1)\};$$
$$C_{\text{вн}_4} = \{9(3,9) - 12(9,4) - 10(4,6) - 8(6,3)\};$$
$$C_{\text{вн}_5} = \{9(3,9) - 12(9,4) - 10(4,6) - 13(6,5) - 7(5,3)\};$$

$$C_{\text{вн}_6} = \{12(9,4) - 10(4,6) - 13(6,5) - 14(5,7) - 17(7,9)\};$$
$$C_{\text{вн}_7} = \{10(4,6) - 13(6,5) - 14(5,7) - 11(7,4)\};$$
$$C_{\text{вн}_8} = \{13(6,5) - 14(5,7) - 16(7,8) - 15(8,6)\}.$$

Алгоритм плоской укладки завершил работу. В результате работы получена укладка максимальной планарной части исходного графа G' (рис. 5.34).

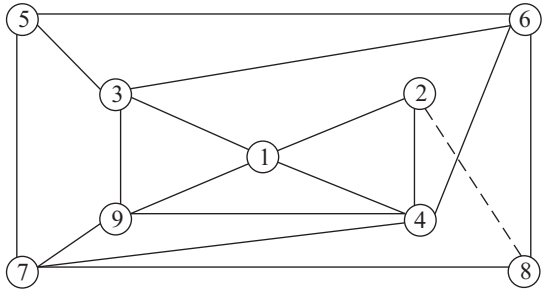


Рис. 5.34. Максимальная планарная укладка графа G'

Для алгоритма определения планарности в классической постановке временная и пространственная сложности определяются как функции от n, m. Временная сложность генетического алгоритма складывается из временных сложностей отдельных операторов. Временная сложность алгоритма селекции в худшем случае составляет $(Np + 2\log 2Np)$. Временная сложность оператора кроссинговера и оператора мутации составляют $O(L)$.

В течение одного поколения выполняется один оператор селекции и один оператор кроссинговера. Оператор мутации выполняются не чаще одного раза в 4–5 поколений. Необходимо учесть, что при начальной генерации популяции формирование одной хромосомы требует L операций для определения генов. После генерации начальной популяции необходимо выполнить ее сортировку, что требует $N_p \log 2N_p$ операций, т. е. временная сложность генетического алгоритма составит

$$O((n+m) + 2m + [N_p \log 2N_p + (N_p + 2\log 2N_p) + O(L) + O(L)/5]N_G),$$

где N_G — число поколений, N_p — размер популяции. Следовательно, временная сложность алгоритма определения планарности пропорциональна $O(n+m)$.

5.5. Определение изоморфизма графов

Большое число комбинаторно-логических инженерных задач на графах требует установления изоморфизма или изоморфного вложения между заданными структурами. Данная проблема, как и все остальные рассмотренные выше проблемы, является NP-полной. Поэтому разра-

батываются различные эвристики для получения приемлемых практических результатов.

Задача распознавания изоморфизма графов состоит в следующем. Для заданных графов $G_1 = (X_1, U_1)$ и $G_2 = (X_2, U_2)$ требуется определить, существует ли взаимно однозначное отображение $\varphi : X_1 \to X_2$ такое, что $U = (x, y) \in U_1$ тогда и только тогда, когда $(\varphi(x), \varphi(y)) \in U_2$.

В настоящее время известны полиномиальные алгоритмы для следующих классов графов: графы ограниченной степени; графы с ограниченной кратностью собственных значений; к-разделимые графы; k-стягиваемые графы и так далее.

Существует ряд задач, которые полиномиально сводятся друг к другу и к задаче распознавания изоморфизма графов G_1 и G_2.

1. Распознавание изоморфизма графов и построение порождающего множества для группы автоморфизмов графа, где $G = G_1 \cup G_2$.
2. Распознавание изоморфизма графов и существование изоморфной подстановки (временная сложность алгоритма $\approx 0(n^2)$).
3. Распознавание изоморфизма графов и число симметрии графа (временная сложность алгоритма $\approx 0(n^2)$).
4. Распознавание изоморфизма графов и автоморфное разбиение множества вершин графа (временная сложность алгоритма $\approx 0(n^2)$).
5. Распознавание изоморфизма графов и число изоморфизмов графов (временная сложность алгоритма $\approx 0(n^2)$).
6. Распознавание изоморфизма графов и определение существования автоморфизма для пары фиксированных вершин графа G (временная сложность алгоритма $\approx 0(n^2)$).

Известно, что задача изоморфного вложения графа также является NP-полной задачей. Она имеет много сходства и в то же время существенно (по сложности) отличается от задачи распознавания изоморфизма графов. Например, для решения задачи изоморфизма подграфа G_1 с использованием известных алгоритмов распознавания изоморфизма графов необходимо разработать процедуру выделения в графе G подмножества $X_1 \subset X$, равномощного с множеством вершин X_2 графа G_2.

Данная процедура включает k_1 действий, где $k_1 = \begin{pmatrix} n \\ n_2 \end{pmatrix}$, $n = |X|$, $n_2 = |X_2|$. Следовательно, k_1 раз надо применять и алгоритм распознавания изоморфизма графов. Поэтому при выделении каждого подграфа G_1 в графе G необходимо выполнять k_2 действий, где $k_2 = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix}$ (m_1 — количество ребер в подграфе G_1, m_2 — в графе G_2, $m_1 > m_2$). Следовательно, даже если есть полиномиальный алгоритм распознавания изоморфизма графов, с его помощью невозможно решить задачу изоморфного вложения за полиномиальное время.

Наибольшую трудность представляет установление изоморфизма однородных графов, имеющих автоморфные подграфы. Для решения

таких задач используются методы разбиения исследуемых графов на различные уровни. При этом временная сложность алгоритма снижается с $O(n!)$ до $O(k!)$, где n — число элементов в графе, а k — число элементов в наибольшем автоморфном подграфе, т. е. в таком подграфе, где не имеет значения выбор вершин для установления соответствия.

Основная идея таких алгоритмов заключается в следующем. В графах, исследуемых на изоморфизм, выбираются две предполагаемо изоморфные вершины. Относительно них производится разбиение всех оставшихся вершин на два класса. В первый класс включаются вершины, смежные выбранным вершинам, а во второй — не смежные.

Например, на рис. 5.35, *а*, *б* показаны два графа G, G'. Пусть $G = (X, U)$, $G' = (X', U')$, $|X| = |X'| = 6$, $|U| = |U'| = 9$, локальные степени всех вершин графа равны трем. Необходимо установить изоморфизм графов G и G'. Другими словами, необходимо определить, существует ли отношение эквивалентности

$X \Leftrightarrow X'$, $U \Leftrightarrow U'$ такое, что,
если ребро $(x_i, x_j) \Leftrightarrow$ ребру (x'_i, x'_j),
тогда $x_i \Leftrightarrow x'_i$, $x_j \Leftrightarrow x'_j$, $(x_i, x_j) \in X$, $(x'_i, x'_j) \in X'$.

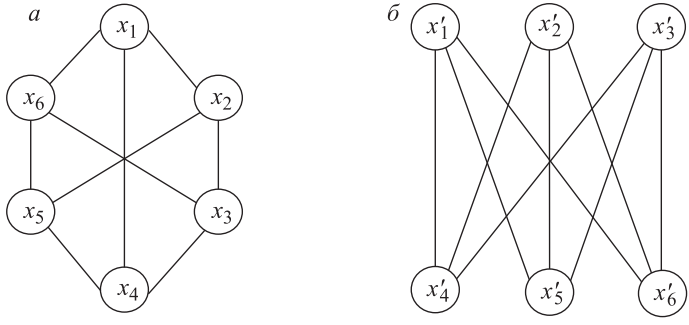


Рис. 5.35. *а* — граф G, *б* — граф G'

Покажем на примере реализацию эвристики разбиения для установления изоморфизма однородных графов. Выберем одну вершину в графе G и одну в графе G' и относительно них выполним указанное выше разбиение:

$$\{1\} \quad \{2,4,6\}_{1+} \quad \{3,5\}_{1-} \quad (G)$$
$$\{1'\} \quad \{4',5',6'\}'_{1+} \quad \{2',3'\}'_{1-} \quad (G')$$

В данном разбиении вершины x_2, x_4, x_6 смежны вершине x_1, а вершины x_3, x_5 не смежны вершине x_1 графа G. Аналогичное разбиение выполнено для графа G'. Здесь считается, что вершины x_1 и x'_1 предполагаемо изоморфны (П-изоморфны). Далее выбираются

подмножества меньшей мощности и внутри них проверяется смежность вершин. В нашем примере вершины x_3, x_5 и x'_2, x'_3 не смежны. Поэтому процесс разбиения продолжается:

$$\begin{array}{llll} \{1\} & \{3\}_{1-}\{\{2,4,6\}_{3+}\}_{1+} & & \{\{5\}_{3-}\}_{1-} \\ \{1'\} & \{2'\}'_{1-}\{\{4',5',6'\}'_{2+}\}'_{1+} & \{\{3'\}'_{2-}\}'_{1-} & \end{array}$$

Продолжая процесс аналогично, получим, что граф G изоморфен графу G'. Подстановка вершин запишется так:

$$t = \left\{ \begin{array}{cccccc} 1 & 2 & 4 & 6 & 3 & 5 \\ 1' & 4' & 5' & 6' & 2' & 3' \end{array} \right\}.$$

В рассматриваемом примере подмножества вершин $\{x_2, x_4, x_6\}$ и $\{x'_4, x'_5, x'_6\}$ являются автоморфными, т. е. каждая вершина в одном подграфе может быть изоморфна любой вершине второго подграфа. В примере (рис. 5.35) временная сложность алгоритма распознавания изоморфизма графов $O(6!)$ была сведена к временной сложности алгоритма распознавания изоморфизма графов $O(3!)$. В алгоритмах такого типа необходим перебор между элементами автоморфных подграфов, причем, как следует из рассмотрения примера, такой перебор здесь выполняется последовательно для всех вершин, кроме последней вершины в подграфе.

Предлагается в рассматриваемом методе распознавания изоморфизма графов после получения подмножеств разбиения в подмножествах наибольшей мощности выполнять все модифицированные генетические операторы. Наилучшие результаты дало применение новых модифицированных генетических операторов, основанных на «жадной» стратегии, методе дихотомии, минимального кластера, методе золотого сечения и методе Фибоначчи. Это позволяет параллельно анализировать все подмножества автоморфных вершин и повышает время поиска.

Например, на рис. 5.36, *а*, *б* показаны нетривиальные, неоднородные графы G и G'. В этих графах

$$|X| = |X'| = 7, \quad |U| = |U'| = 8,$$

$$\rho_1 = 1, \quad \rho_2 = 3, \quad \rho_3 = 3, \quad \rho_4 = 3, \quad \rho_5 = 3, \quad \rho_6 = 2, \quad \rho_7 = 1;$$
$$\rho_a = 1, \quad \rho_b = 3, \quad \rho_c = 3, \quad \rho_d = 3, \quad \rho_e = 3, \quad \rho_f = 2, \quad \rho_g = 1.$$

Вершина 1 может быть П-изоморфна вершине a или g. В первом случае нас ждет тупик, а во втором — успех. Мы предположили, что вершина 1 П-изоморфна вершине g. Тогда на основе связности графа и эвристики разбиения следует, что вершина 7 П-изоморфна вершине a. После этого вытекает, что вершина 6 П-изоморфна вершине f. Далее необходимо установить, существует ли соответствие между подмножествами автоморфных вершин $\{2, 3, 4, 5\}$ и $\{b, c, d, e\}$. В этом случае, как отмечалось выше, необходимо выполнить полный перебор между

этими подмножествами вершин. Для этих подмножеств это — 4!. Выполняя такие преобразования, получим, что вершина 2 П-изоморфна вершине b, вершина 3 П-изоморфна вершине d, 4 — c и 5 — e. Отсюда следует, что графы G и G' (рис. 5.36, $а$, $б$) изоморфны, а подстановка изоморфизма, переводящая один граф в другой, запишется так:

$$t = \left\{ \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ a & b & d & c & e & f & g \end{matrix} \right\}.$$

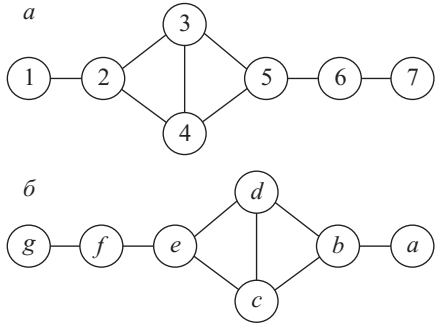


Рис. 5.36. $а$ — нетривиальный граф G, $б$ — нетривиальный граф G'

Как видно, наличие вершин с локальными степенями различной мощности упрощает процесс распознавания изоморфизма графов. Это позволяет выбирать начальные условия для эвристики разбиения. В примере (рис. 5.36, $а$, $б$) всего два способа для выбора разбиения, в то время как в графах (рис. 5.35, $а$, $б$) шесть возможных способов разбиения. В однородных графах все степени вершин одинаковы, поэтому начальные условия выбираются произвольно случайным образом. В этой связи при установлении изоморфизма в однородных графах временная сложность алгоритма в самом лучшем случае $O(n)$, в самом худшем случае $O(n!)$. Если графы неизоморфны, то за одну итерацию установить результат невозможно. Необходимо провести сравнение на изоморфизм одной случайно выбранной вершины из одного графа со всеми остальными вершинами другого графа.

Например, пусть заданы два графа (рис. 5.37, $а$, $б$) G и G'. При этом

$$G = (X, U), \quad G' = (X', U'), \quad |X| = |X'| = 6, \quad |U| = |U'| = 9$$

— локальные степени всех вершин графа равны трем.

Для распознавания изоморфизма графов применим эвристику разбиения. Предположим, что вершина 1 графа G П-изоморфна вершине a графа G'. Тогда получим

$$\{1\} \quad \{2,4,6\}_{1+} \quad \{3,5\}_{1-} \quad (G)$$
$$\{a\} \quad \{b,d,f\}_{1'+} \quad \{e,c\}_{1'-} \quad (G')$$

Для дальнейшего анализа выбираем соответствующие подмножества наименьшей мощности $\{3,5\}$ и $\{e,c\}$. Вершины $(3,5)$ в графе G (рис. 5.37, a) смежны, а вершины (c,e) в графе G' — нет. Следовательно, вершина 1 не может быть изоморфна вершине a.

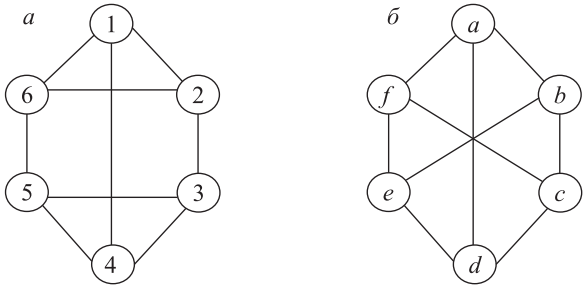


Рис. 5.37. a — граф G, $б$ — граф G'

В этой связи необходимо провести аналогичные операции для проверки изоморфизма вершины 1 с остальными вершинами графа G'. Далее получим

$$\{1\} \quad \{2,4,6\}_{1+} \quad \{3,5\}_{1-} \quad (G)$$
$$\{b\} \quad \{a,c,e\}_{1'+} \quad \{d,f\}_{1'-} \quad (G')$$

Вершины $(3,5)$ в графе G (рис. 5.37, a) смежны, а вершины (d,f) в графе G' — нет. Следовательно, вершина 1 не может быть изоморфна вершине b. Продолжая аналогичные построения, имеем, что вершина 1 не может быть изоморфна вершинам c, d, e. Наконец, предположим, что вершина 1 графа G П-изоморфна вершине f графа G'. Тогда получим

$$\{1\} \quad \{2,4,6\}_{1+} \quad \{3,5\}_{1-} \quad (G)$$
$$\{f\} \quad \{a,c,e\}_{1'+} \quad \{b,d\}_{1'-} \quad (G')$$

Вершины $(3,5)$ в графе G (рис. 5.37, a) смежны, а вершины (b,d) в графе G' — нет. Следовательно, вершина 1 не может быть изоморфна вершине b. Проанализированы все вершины графа G' на предмет изоморфизма с вершиной 1 графа G. Во всех случаях ответ отрицательный. Поэтому граф G неизоморфен графу G'.

Рассмотрим пример распознавания изоморфизма графов (рис. 5.38, a, $б$) G и G'. При этом $G = (X,U)$, $G' = (X',U')$, $|X| = |X'| = 10$, $|U| = |U'| = 15$, локальные степени всех вершин графа равны трем. Предположим, что вершина 1 графа G П-изоморфна вершине a графа G'. Тогда получим

$$\{1\} \quad \{2,5,6\}_{1+} \quad \{3,4,7,8,9,10\}_{1-} \quad (G)$$
$$\{a\} \quad \{b,e,f\}_{1'+} \quad \{c,d,h,g,k,i\}_{1'-} \quad (G')$$

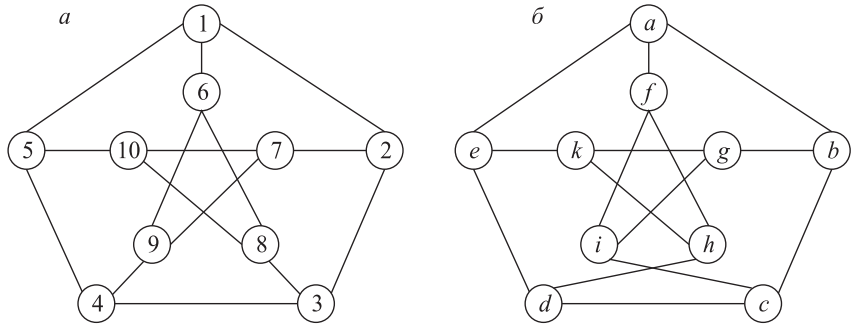


Рис. 5.38. $а$ — граф G, $б$ — граф G'

Для дальнейшего анализа выбираем соответствующие подмножества наименьшей мощности $\{2,5,6\}$ и $\{b,e,f\}$. Предположим, что вершина 2 П-изоморфна вершине b. В этом случае получим

$$\{1\}\ \ \{2\}_{1+}\ \ [\{5,6\}_{1+}]_{1-}\ \ [\{7,3\}_{1+},\{,4,8,910\}_{1-}]_{1-}\ \ (G)$$
$$\{a\}\ \ \{b\}'_{1+}\ \ [\{e,f\}'_{1+}]'_{1-}\ \ [\{g,c\}'_{1+},\{d,h,k,i\}'_{1-}]'_{1-}\ \ (G')$$

Как видно, П-изоморфизм сохраняется. Продолжая далее, запишем

$$\{1\}\ \{2\}_{1+}\ [\{5,6\}_{1+}]_{1-}\ [[\{7\}_{1+},\{3\}_{1+}]_{1+},[\{9,10\}_{1-},\{4,8\}_{1-}]_{1-}]_{1-}\ (G)$$
$$\{a\}\ \{b\}'_{1+}\ [\{e,f\}'_{1+}]'_{1-}\ [[\{g\}'_{1+},\{c\}'_{1+}]'_{1+},[\{i,k\}'_{1-},\{h,d,\}'_{1-}]'_{1-}]'_{1-}\ (G')$$

Так как вершины (h,d) в графе G' (рис. 5.38, $б$) смежны, а вершины $(4,8)$ в графе G (рис. 5.38, $а$) — нет, то вершина 1 не может быть изоморфна вершине a, вершина 2 — b, вершина 7 — g и вершина 3 — c. Продолжая аналогично, получим, что граф G не изоморфен графу G'.

Предлагается подход на основе микро-, макро-, и метаэволюции для нахождения подстановки изоморфизма графов, если она существует. На этапе микроэволюции эвристика разбиения позволяет получить строительные блоки. Перераспределение генетического материала на основе генетических операторов и их модификаций выполняется внутри каждого строительного блока. За счет этого уменьшается размер хромосом в популяции и сокращается время реализации основного алгоритма. Например, в хромосоме $P_1 : 2,3,4,5$, которая является строительным блоком для графа G (рис. 5.37, $а$), определим точку мутации между 3 и 4 геном. Выполнив оператором мутации, получим хромосому-потомок $P_2 : 2,4,3,5$. Она определяет подстановку 2-b, 4-c, 3-d и 5-e.

На уровне макроэволюции эффективно использовать фрактальные множества, при этом каждый строительный блок представляется как объединенная вершина нового графа. В этом случае размер хромосом уменьшается и появляется возможность увеличить число генераций

генетического алгоритма для получения оптимального результата. Это особенно актуально при анализе однородных неизоморфных графов с одинаковым числом вершин и ребер.

На этапе метаэволюции происходит миграция хромосом из одной популяции в другую и различные модифицированные генетические операции над популяциями.

Приведем структурную схему генетического поиска для решения переборных комбинаторно логических задач на графах (рис. 5.39) на основе информирующих обратных связей и концепции объединенной эволюции. После реализации генетического алгоритма на рис. 5.39 компенсатор при взаимодействии с внешней средой реализует синергетические принципы, а фильтр хромосом поддерживает гомеостаз. При этом лучшие хромосомы отправляются для смешивания популяций и выхода из локальных оптимумов. Редуктор уменьшает размер популяции, устраняя хромосомы со значением целевой функции ниже средней. Блоки сумматор, редуктор и фильтр хромосом позволяют

Рис. 5.39. Структурная схема генетического поиска

повысить эффективность реализации эволюции и скорость распознавания изоморфизма графов. Следует отметить, что в графах большой размерности с нетривиальными автоморфизмами ($K > 100$) процесс установления изоморфизма резко усложняется, но использование таких схем поиска на порядок снижает временную сложность алгоритма.

Приведенные схемы управления генетическим поиском позволяют повысить качество решения и уменьшить время поиска. Временная сложность алгоритма такого класса лежит в пределах $O(n^2) \div O(n^3)$.

5.6. Генетический алгоритм определения паросочетаний графа

Пусть $G = (X, U)$ — неориентированный граф. ***Паросочетанием*** (ПС) называется подмножество ребер $M \subseteq U$, не имеющих общих концов, причем каждое ребро $u_i \in U$ смежно одному ребру из M. ***Максимальное паросочетание*** — это паросочетание M, содержащее максимально возможное число ребер. Известно, что число ребер в паросочетании графа $G = (X, U)$ $|X| = n$ не превышает $[n/2]$, где $[n/2]$ ближайшее большее целое.

Рассмотрим основные эвристики определения паросочетаний в двудольных графах $G = (X_1 \cup X_2, U)$, $X_1 \cup X_2$, $X_1 \cap X_2 = \varnothing$.

Пусть задан двудольный граф, показанный на рис. 5.40. В нем можно определить паросочетание $M_1 = \{(1,6),(3,8)\}$. В этом графе максимальное паросочетание (МПС) $M_2 = \{(5,8)(3,7)(2,6)\}$, выделенное жирными линиями, показано на рис. 5.41. В этом графе можно построить еще одно МПС: $M_2' = \{(4,8)(3,7)(2,6)\}$. Для нахождения МПС в двудольном графе будем использовать специальную матрицу смежности R:

$R =$

	6	7	8	9
1	1			
2	1		1	
3		1	1	1
4			1	
5			1	

Строки матрицы соответствуют вершинам X_1, а столбцы — вершинам X_2. На пересечении строк и столбцов ставится значение 1 или 0 в зависимости от наличия или отсутствия соответствующего ребра. Такая модель требует в 4 раза меньше ячеек памяти для представления матрицы в ЭВМ. Это особенно существенно при анализе графов на сотни тысяч вершин.

Предлагается следующая эвристика. В матрице R ищется диагональ с наибольшим числом элементов. Если таких диагоналей несколько, то выбирается любая. Например, в матрице R диагональ с наибольшим числом элементов имеет вид Д $= \{(2,6)(3,7)(4,8)\}$. Следовательно, определено МПС M_2' для графа рис. 5.41.

Сформулируем следующую гипотезу. Главная диагональ матрицы R, полностью заполненная элементами, соответствует МПС. Суммарное число единиц соответствует суммарному числу ребер МПС. Каждая единица главной диагонали определяет ребро МПС.

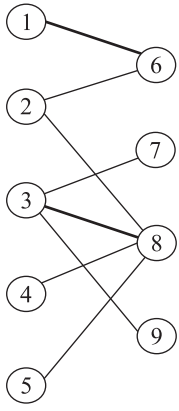


Рис. 5.40. Двудольный граф G

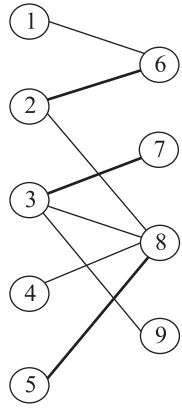


Рис. 5.41. Максимальное ПС

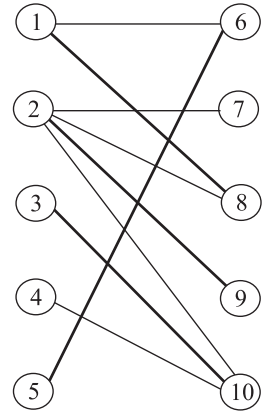


Рис. 5.42. Двудольный граф G

Доказательство следует из способа построения специальной матрицы по заданному двудольному графу, так как каждой вершине из X_1 главной диагонали ставится в соответствие одна и только одна вершина из X_2. Отметим, что перед работой алгоритма необходимо упорядочить вершины двудольного графа по возрастанию элементов.

Опишем основную стратегию определения МПС в двудольном графе.

1. Определить вершины подмножеств X_1 и X_2 двудольного графа G.
2. Упорядочим вершины X_1 и X_2.
3. Построить специальную матрицу R и определить в ней главную диагональ.
4. Если главная диагональ заполнена элементами полностью, то построено МПС и перейти к шагу 7. Если нет, то перейти к 5.
5. В матрице R определить все диагонали и выбрать диагональ с наибольшим числом элементов. Если таких диагоналей несколько, то выбрать любую.
6. Выполнить процедуру склеивания и преобразования на основе генетических операторов. В результате строится МПС.
7. Конец работы алгоритма.

Например, дан двудольный граф рис. 5.42. Построим специальную матрицу R этого графа:

$R =$

	6	7	8	9	10
1	1		1		
2		1	1	1	1
3					1
4					1
5	1				

Согласно алгоритму определяем главную диагональ $Д = \{(1,6)(2,7)\}$. Она не заполнена элементами полностью. Определяем другие диагонали:

$$Д_1 = \{(1,8)(2,9)(3,10)\};\ \ Д_2 = \{(2,10)\};$$

$$Д_3 = \{(2,8)(4,10)\};\ \ Д_4 = \{(5,6)\}.$$

В качестве базовой для определения МПС выбираем $Д_1$ как диагональ, содержащую наибольшее число элементов. Выполняем процедуру склеивания:

$$Д_1 \cap Д_2 \neq \varnothing,\ \ Д_1 \cap Д_3 \neq \varnothing,\ \ Д_1 \cap Д \neq \varnothing,\ \ Д_1 \cap Д_4 = \varnothing.$$

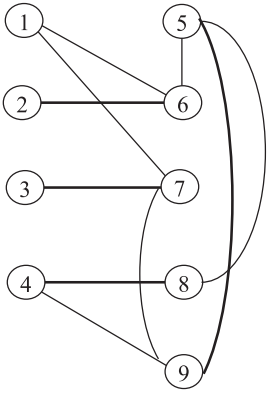


Рис. 5.43. Граф G

Следовательно, элемент $(5,6)$ из $Д_4$ добавляется к $Д_1$. На этом построение МПС завершено: $M_1 = \{(1,8)(2,9)(3,10)(5,6)\}$ и $|M_1| = 4$.

Данная методика может быть применена и для графов, не являющихся двудольными. Для этого необходимо в графе выделить максимально двудольную часть. Для нее определить МПС, а затем на основе склеивания и применения ГО построить МПС для произвольного графа.

Например, пусть задан граф G (рис. 5.43). Для его двудольной части МПС равно $M = \{(2,6)(3,7)(4,8)\}$. Применяя процедуру склеивания, получим МПС графа $M = \{(2,6)(3,7)(4,8)(5,9)\}$.

Отметим, что для построения МПС можно использовать жадную стратегию. Пример такого построения имеет вид:

$1°\ (1,6)$
$2°\ (1,6)(3,7)$
$3°\ (1,6)(3,7)(4,8)$
$4°\ (1,6)(3,7)(4,8)$
$5°\ (1,6)(3,7)(4,8)(5,6)$
$6°\ (1,6)(3,7)(4,8)(5,8)$
$7°\ (1,6)(3,7)(4,8)(5,9)$

В результате построено МПС, $M = \{(1,6)(3,7)(4,8)(5,9)\}$, $|M| = 4$.

Для графов $n < 100$ можно построить семейство всех ПС и выбрать из них наибольшее. Например:

$1\ (1,7)$
$2\ (1,7)(2,6)$
$3\ (1,7)(2,6)(4,8)$
$4\ (1,7)(2,6)(4,8)(5,9)$

Построено новое МПС $M_1 = \{(1,7)(2,6)(4,8)(5,9)\}$, $|M| = 4$.

Применим жадные ГО для графа (рис. 5.43), длина хромосомы в котором $n/2 = 9/2 = 4{,}5$. Следовательно, $L = 5$. Сгенерируем случайным образом популяцию как набор хромосом, состоящих из различных ребер графа $P = \{P_1, P_2, P_3, P_4\}$:

$$P_1 = (1,6)(2,6)(3,7)(4,8)(5,9),$$
$$P_2 = (2,6)(4,9)(5,8)(3,7)(1,7),$$
$$P_3 = (7,9)(5,8)(1,7)(2,6)(5,6),$$
$$P_4 = (4,9)(5,9)(2,6)(4,8)(3,7).$$

Применяя оператор сегрегации из P_1, выбираем подмножество ребер (строительный блок) $\text{СБ}_1 = \{(4,8)(5,9)\}$, из P_2 выбирается $\text{СБ}_2 = \{(1,7)\}$, из P_3 выбирается $\text{СБ}_3 = \{(2,6)\}$. На этом формирование МПС закончено: $M = \{(1,7)(2,6)(4,8)(5,9)\}$. Временная сложность алгоритма в лучшем случае равна $O(nm)$, в самом худшем случае — $O(n!)$. В среднем для реальных графов имеем $O(n^2) \div O(n^3)$, где n — число вершин, а m — число ребер графа.

5.7. Квантовый алгоритм построения Гамильтонова цикла

В последнее время появились новые подходы решения NP-полных проблем, основанные на методах квантового поиска. Квантовый поиск анализирует неструктурированные проблемы, которые в общем виде формулируются следующим образом. Задана функция $f(x)$, аргументы x — целые числа, $x = 1, 2, \ldots, N$, причем $f(x)$ принимает значение ноль во всех случаях, кроме $x = w$. Необходимо найти значение w, используя наименьшее число запросов к $f(x)$. Задачи такого типа при небольшом $x < 100$ решаются на основе полного перебора (исчерпывающего поиска) или методом проб и ошибок.

Идею и структуру квантового алгоритма предложил Л. Гровер. При решении неструктурированной проблемы поиска существует «оракул», определяющий, является ли рассматриваемое решение искомым.

Для реализации поиска это квантовое пространство развивается в общую суперпозицию, которая концентрируется в векторе $\mathbf{t}$, определяющем путь до цели поиска. Предлагается процедура квантового кругооборота U. Л. Гровер предлагает использовать U и $f(x)$, чтобы построить увеличивающий амплитуду оператор Q, который изменяет амплитуду вероятности от не-цели векторов $\mathbf{S} \neq \mathbf{t}$ в цель $\mathbf{S} = \mathbf{t}$. Поведение «оракула» в алгоритме квантового поиска моделируется возвратной функцией $f(x) = 0$, для всех x, w и $f(x) = 1$, для $x = w$.

Для решения NP-полных проблем на графах предлагается анализировать структуру графа, чтобы «выращивать» полные решения, рекурсивно расширяя последовательные частичные решения.

В 1859 г. Вильям Гамильтон сформулировал задачу нахождения пути вдоль ребер додекаэдра, проходящего по одному разу через каждую

вершину многоугольника и возвращающегося в исходную точку. Позже цикл, проходящий по одному разу все вершины графа, был назван *гамильтоновым циклом* (ГЦ); путь, проходящий через все вершины графа — *гамильтоновым путем* (цепью), а граф, содержащий ГЦ, был назван *гамильтоновым графом*. Задачи нахождения гамильтоновых циклов и цепей находят применение во многих практических задачах кибернетики, управления, экономики, искусственного интеллекта и т. п. До сих пор не найдены необходимые и достаточные условия существования гамильтоновых циклов в неориентированных графах. В этой связи разрабатываются различные эвристики решения данной задачи. Считается, что проблема определения ГЦ в графе является NP-полной, т. е. не существует общего эффективного алгоритма решающего эту задачу за полиномиальное время. В то же время были найдены эффективные эвристики, решающие частные задачи этого класса. Каждый раз с появлением новой технологии решения комбинаторно-логических задач на графах возникают новые эвристики решения такого рода задач. В настоящее время известны эвристики определения ГЦ в графе на основе таких методов, как моделирование отжига, генетические алгоритмы, эволюционные стратегии.

Опишем ряд эвристик определения ГЦ на основе квантовых алгоритмов и жадных стратегий. Это позволяет в определенных случаях получать алгоритмы с полиномиальной временной сложностью.

Приведем модифицированный алгоритм квантового поиска, ориентированный на решение задачи определения гамильтонова цикла в графе.

1. Начало.
2. Ввод исходных данных.
3. Проверка необходимых условий существования ГЦ в графе.
4. Анализ математической модели и на его основе построение дерева частичных решений.
5. Суперпозиция частичных решений на основе жадной стратегии и квантового поиска.
6. В случае наличия тупиковых решений — последовательный поиск с пошаговым возвращением.
7. Если набор полных решений построен, то переход к 7, если нет, то к 5.
8. Лексикографический перебор полных решений и выбор из него оптимального или квазиоптимального решения.
9. Конец работы алгоритма.

Приведем укрупненный код алгоритма квантового поиска гамильтонова цикла в графе на Паскале.

```
begin {основная программа}
 generation:=0 {установка}
  initialize;
  repeat {основной цикл}
    gen:=gen+1
```

```
    generation;
    cycle;
    return;
statistics (max, avg, min, sumfitness, newsol)
report (gen)
   oldsol:=newsol
   until (gen >= maxgen)
end {конец основной программы}.
```

Здесь `generation (gen)` — генерации (итерации) алгоритма; `max`, `avg`, `min`, `sumfitness` — максимальное, среднее, минимальное, суммарное значение целевой функции; `oldsol`, `newsol` — старое и новое решение соответственно. Алгоритм квантового поиска (1 итерация выполняется в блоке `repeat`).

Работа алгоритма начинается с чтения данных, инициализации случайных решений, вычисления статистических данных и их печати. Процедура `report` представляет полный отчет обо всех параметрах алгоритма. На основе анализа данных из процедуры `report` строится график зависимости значений целевой функции от числа генераций. Если функция имеет несколько локальных оптимумов и мы попали в один из них, то увеличение числа генераций может не привести к улучшению значений целевой функции. В этом случае наступила предварительная сходимость алгоритма. Операция return предусматривает пошаговый возврат до выхода из тупика. Алгоритмы квантового поиска весьма чувствительны к изменениям и перестановкам входных параметров исходной модели. Это говорит о том, что, например, для одного вида модели объекта, представленного матрицей, можно получить решение с одним локальным оптимумом. А для этой же матрицы с переставленными строками и столбцами можно получить другое решение с лучшим локальным оптимумом. Следует отметить, что, изменяя параметры, алгоритмы и схему квантового поиска, в некоторых случаях можно выходить из локальных оптимумов. Эта проблема продолжает оставаться одной из важнейших во всех методах оптимизации.

5.8. Нечеткие генетические алгоритмы решения задач оптимизации и проектирования

1. При решении многих задач оптимизации, проектирования и принятия решений часто приходится встречаться с таким понятием, как неопределенность в отношении, какого-либо параметра. Для решения данной проблемы был предложен новый подход на основе создания гибридных систем и алгоритмов на основе интеграции нечетких математических моделей и методов эволюционного и генетического поиска. Одним из таких методов являются нечеткие генетические, эволюционные и адаптивные алгоритмы.

Как известно, генетические алгоритмы представляют собой адаптивный поисковый метод, который основан на селекции лучших элементов в популяции, подобно эволюционной теории Ч. Дарвина.

Впервые генетические алгоритмы были применены к решению таких научных проблем, как распознавание образов и оптимизация. Основой для возникновения генетических алгоритмов послужила модель биологической эволюции и методы случайного поиска, т. е. последовательное преобразование одного конечного нечеткого множества промежуточных решений в другое.

Генетические алгоритмы эффективно используют информацию, накопленную в процессе эволюции.

Суть генетических алгоритмов состоит в моделировании естественных эволюционных процессов для эффективного решения оптимизационных задач науки и техники.

В настоящее время используется новая парадигма решения оптимизационных задач на основе генетических алгоритмов и их различных модификаций. Они осуществляют поиск баланса между эффективностью и качеством решений за счет «выживания сильнейших альтернативных решений» в неопределенных и нечетких условиях.

Для работы генетических алгоритмов выбирают множество натуральных параметров оптимизационной проблемы и кодируют их в последовательность конечной длины в некотором алфавите. Они работают до тех пор, пока не будет выполнено заданное число генераций (итераций алгоритма) или на некоторой генерации будет получено решение определенного качества, или, когда найден локальный оптимум, происходит возникновение преждевременной сходимости и алгоритм не может найти выход из этого состояния. Генетический алгоритм, как правило, анализирует различные области пространства решений одновременно и поэтому они более приспособлены к нахождению новых областей с лучшими значениями целевой функции.

Мягкие вычисления — сложная компьютерная методология, основанная на нечеткой логике, генетических вычислениях, нейрокомпьютинге и вероятностных вычислениях. Составные части не конкурируют, но создают эффект взаимного усиления для достижения робастности, низкой цены решения, повышения эффективности приложений. Можно выделить четыре наиболее крупные составные части направления «мягкие вычисления»:

- нечеткая логика (приближенные вычисления, грануляция информации, вычисление на словах);
- нейрокомпьютинг (обучение, адаптация, классификация, системное моделирование и идентификация);
- генетические вычисления (синтез, настройка и оптимизация с помощью систематизированного случайного поиска и эволюции);
- вероятностные вычисления (управление неопределенностью, сети доверия, хаотические системы, предсказание).

Математический аппарат теории нечетких систем используется в данном случае для кодирования, подбора оптимальных параметров генетических алгоритмов, значений вероятности генетических опера-

торов, выбора функции пригодности, создания нечетких генетических операторов. Есть два преимущества нечеткого кодирования:

1) кодовые последовательности могут быть неоднородными и ориентироваться на отдельные многообещающие области поиска, что позволит сократить область поиска и соответственно вычислительные затраты; в закодированную последовательность может быть неявным образом включена функция пригодности;

2) нечеткое кодирование позволяет выполнять так называемое слабое кодирование оптимизируемых структур.

В настоящее время наибольшие успехи в интеграции систем и подходов нечеткой логики и генетических алгоритмов (ГА) достигнуты в следующих двух областях:

1) применение механизмов генетических и эволюционных алгоритмов для решения проблем оптимизации и поиска в условиях нечеткой, неопределенной или недостаточной информации об объекте, параметрах и критериях решаемой задачи, совместно с использованием систем нечеткой логики;

2) использование нечетких инструментов и методов, основанных на нечеткой логике для моделирования различных компонентов и операторов генетических алгоритмов, а также для адаптации и управления основными параметрами генетического алгоритма для динамической настройки и улучшения работы ГА.

Как правило, под нечетким генетическим алгоритмом (НГА) понимают гибридные структуры, относящиеся ко второй области. Таким образом, нечеткий генетический алгоритм можно определить как алгоритм, сочетающий поисковые возможности генетических алгоритмов и возможности математического аппарата нечеткой логики. Математический аппарат теории нечетких систем используется в данном случае для кодирования, подбора оптимальных параметров генетических алгоритмов, значений вероятности генетических операторов, выбора функции пригодности и критерия останова, создания нечетких генетических операторов.

Рассмотрим подробнее основные направления создания и модификации НГА.

• ***Контроль и адаптация основных управляющих параметров генетического алгоритма.*** Для контроля и динамического изменения соответствующих параметров генетического алгоритма в систему вводится *нечеткий логический контроллер* (НЛК), который, используя опыт и знание экспертов в рассматриваемой области, соответствующим образом динамически изменяет параметры генетического поиска в ходе выполнения ГА для того, чтобы избежать проблемы преждевременной сходимости.

В составе НЛК можно выделить следующие блоки:

- база знаний, включающая в себя базу правил и базу данных;
- блок фаззификации;

- блок дефаззификации;
- система вывода решения;
- система контроля.

В общем, схему работу НЛК можно описать следующим образом. НЛК использует знания экспертов в форме лингвистических правил контроля. Система выработки правил на основе знаний экспертов и использование рассуждений влекут определенный вывод, который после дефаззификации превращается из нечеткого правила в реальное воздействие на параметры алгоритма. Изменение параметров алгоритма влечет за собой изменение процесса поиска и текущих результатов, которые затем в блоке фаззификации из переменных состояния преобразуются в нечеткие множества. Тогда обобщенную структуру логического контроллера можно представить следующим образом (рис. 5.44).

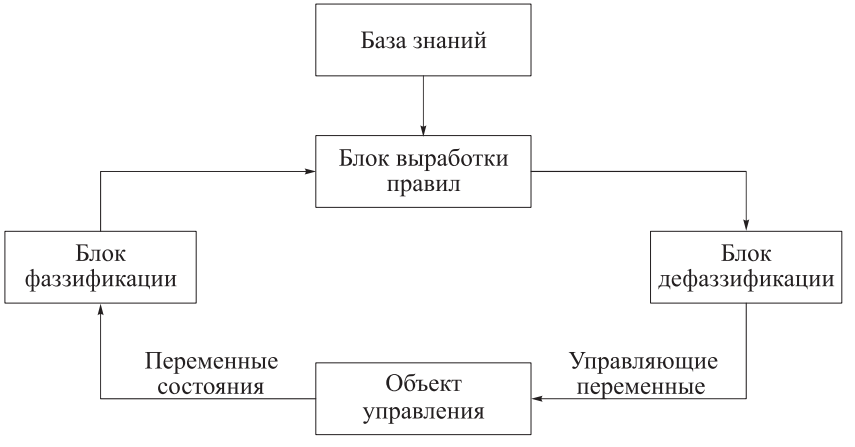


Рис. 5.44. Обобщенная структура нечеткого логического контроллера

База правил, иногда называемая лингвистической моделью, представляет собой множество нечетких правил. НЛК оперирует нечеткими множествами. Поэтому значение входного параметра НЛК подлежит операции *фаззификации*, в результате которой ему будет сопоставлено нечеткое множество.

В соответствии с обобщенным нечетким правилом modus ponens на выходе блока выработки решения формируется либо одно, либо N нечетких множеств. Тогда отображение этих нечетких множеств (или множества) в единственное значение, которое представляет собой управляющее воздействие на объекты управления НЛК, называется дефаззификацией (*defuzzification*). Наиболее распространенными являются следующие методы дефаззификации:

1. Метод дефаззификации по среднему центру (*center average defuzzificatian*).

2. Метод дефаззификации по сумме центров (*center of sums defuzzification*).
3. Метод дефаззификации по центру тяжести (*center of gravity method*, *center of area method*).
4. Метод дефаззификации по максимуму функции принадлежности.

Очевидно, что эффективность работы НГА напрямую зависит от качества (степени соответствия поставленной задачи) разработанных блоков НЛК.

В генетических алгоритмах (ГА) управляющими параметрами, как правило, являются значения вероятности выполнения генетических операторов кроссинговера (P_c) и мутации (P_m), а также размер популяции. В нечетком генетическом алгоритме происходит динамическое изменение этих параметров при помощи нечеткого логического контроллера (НЛК).

Известно, что эффективность работы генетического алгоритма напрямую зависит от выбора его параметров. Так, при большой вероятности кроссинговера увеличивается вероятность уничтожения решений, имеющих высокие значения функции пригодности. Низкая вероятность кроссинговера может затруднить получение лучших решений и не гарантирует более быструю конвергенцию. Высокая вероятность мутации «хороших» решений увеличивает разнообразие популяции и соответственно время решения. Низкая вероятность мутации может приводить к пропуску некоторых, близких к оптимальным, решений. Использование нечетких логических контроллеров для изменения параметров генетического алгоритма позволяет улучшить работу генетического алгоритма за счет более осторожного, взвешенного и целенаправленного контроля. Возможен вариант, когда вероятности кроссинговера и мутации будут определяться НЛК исходя из оценки не всей популяции, а по определенной выборке решений, учитывающей значения функции пригодности и разнообразие популяции, также могут использоваться несколько НЛК.

• ***Разработка и модификация операторов кроссинговера.*** Нечеткие связки и треугольные функции распределения вероятности можно использовать для создания эффективных операторов кроссинговера, которые устанавливают адекватные уровни разнообразия популяции и таким образом позволяют решать проблему преждевременной сходимости.

При решении задач оптимизации функций в непрерывных областях поиска, одной из проблем является обеспечение высокой точности получаемых решений. Для решения данной проблемы были предложены генетические операторы, адаптированные для хромосом с вещественным кодированием. Такие операторы позволяют выполнять локальную настройку решений. В качестве примера такого оператора — можно привести оператор неоднородной мутации, который уменьшает интенсивность мутации генов по мере продвижения к оптимуму. Эта

особенность данного оператора мутации приводит к тому, что в начальной области выполняется практически однородный поиск, а затем в отдельных локальных областях характер поиска варьируется. Также существуют меметические алгоритмы с вещественным кодированием (*real-coded memetic algorithms — RCMA*), которые используют механизмы локального поиска для того, чтобы эффективно очистить решения. Обычно в таких случаях RCMA используют методы градиента или скорейшего спуска–подъема, которые помогают находить локальные оптимумы. Часто в RCMA методы локального поиска применяют к решениям в популяции после выполнения операторов рекомбинации и мутации с целью исследования лучших областей пространства поиска, полученных в ходе выполнения итераций генетического алгоритма.

Можно также отметить еще одну разновидность RCMA, где используются алгоритмы локального поиска на основе кроссинговера (*crossover-based local search algorithms — XLS*). Так как оператор кроссинговера производит решения-потомки вокруг родительских решений, это свойство может быть использовано в качестве оператора движения для методов локального поиска, что представляет интерес при вещественном кодировании. По мере смены поколений популяция теряет разнообразие. Это позволяет кроссинговеру создавать потомство, плотно распределенное вокруг родителей, стимулирует эффективную локальную настройку. Этот вид оператора кроссинговера дает надежду на построение эффективных локальных поисковых методов на базе XLS.

В работе описывается большое количество различных операторов кроссинговера для генетических алгоритмов с вещественным кодированием.

1). Унимодальный оператор кроссинговера с нормальным распределением (*unimodal normal distribution crossover — UNDX*) на основе множества родителей создает решения-потомки, распределенные вокруг центра массы этих родителей. При этом чем дальше решение находится от центра массы, тем меньшая вероятность ему назначается.

2). Симплексный кроссинговер (SPX) назначает однородное распределение вероятности для того, чтобы создать потомство в ограниченном пространстве поиска вокруг области занятой родителями.

Они также называются операторами усредненного центра; их использовали при построении специализированной модели генетического поиска или модель минимальных промежутков в поколениях (*MGG model*). В такой модели из нескольких родительских решений создаются 200 решений-потомков, из которых отбираются два лучших решения.

3). Родительски центрированный оператор рекомбинации, который присваивает тем большую вероятность создаваемым решениям-потомкам, чем ближе они находятся к родителям в пространстве поиска.

Можно также упомянуть еще ряд операторов кроссинговера: смешивания; имитирующий бинарный; арифметический; промежуточный;

расширенный, которые можно рассматривать в качестве разновидностей перечисленных выше операторов.

4). Нечеткие операторы кроссинговера и мутации на основе логических операций. Новые решения образуются из уже имеющихся за счет использования логических операций И, ИЛИ, НЕ, И–НЕ, ИЛИ–НЕ. Например, оператор кроссинговера построен на основе операции ИЛИ–НЕ.

В литературе также описано значительное количество других нечетких генетических операторов: нечеткий кроссинговер, основанный на связях; кроссинговеры с использованием множества родителей; образующий множество потомков и др.

• ***Кодирование решений.*** Классическое бинарное представление решений, когда гены принимают значения ноль или единица, может быть, преобразовано в нечеткое представление, где гены принимают значения в интервале между нулем и единицей). Это позволит выражать более сложные особенности как генотипа, так и фенотипа различных решений популяции наподобие тех, которые встречаются в природе.

Кодирование на основе вещественных чисел наиболее эффективно при решении задач параметрической оптимизации с переменными, заданными в непрерывных областях значений. В таком кодировании хромосома обычно представляет собой вектор чисел с плавающей запятой, причем размер чисел соответствует длине вектора, который является решением рассматриваемой проблемы [5]. Генетические алгоритмы, основанные на вещественном представлении чисел, называют генетическими алгоритмами с вещественным кодированием (*real-coded GAs — RCGA*).

• ***Критерии остановки.*** Такие характеристики нечеткой логики, как степень истинности, оценка достоверности той или иной логической функции, могут быть использованы для получения оптимальных решений с заданной пользователем точностью, прогнозирования и регулирования критериев останова генетического алгоритма по мере достижения заданных значений.

2. *Построение временного графика процесса проектирования на основе нечеткого генетического алгоритма.* Разнообразие конструктивно-технологических методов создания сверхбольших интегральных схем обусловлено стремлением не только улучшить их технико-экономические показатели, но и достичь общих целей: минимизировать длительность процесса проектирования, обеспечить проектирование сверх больших интегральных схем высокой сложности, повысить качество проектирования (главным образом, безошибочность).

Нечеткие системы позволяют решать задачи оптимальности с нечеткими или неточными параметрами, находя решение, наиболее оптимальное для данной задачи. Применение нечетких систем в современных средствах проектирования, в различных отраслях

промышленности при выполнении проектных работ дало возможность повысить производительность труда в несколько десятков раз.

Нечеткие генетические алгоритмы с успехом применяют в современных средствах автоматизации проектирования изделий электронной техники, которая содержит сложнейшие электронные системы, содержащие миллионы электронных компонентов.

Проектирование СБИС представляет собой многоуровневый процесс, в котором каждый уровень характеризуется своим математическим и программным обеспечением. В функциональном проектировании выделяют системный, функционально-логический (регистровый и логический), схемотехнический и компонентный уровни. Как правило, используется нисходящий стиль проектирования, при котором последовательно выполняются процедуры системного, регистрового и логического уровней.

После получения результатов функционально-логического проектирования приступают к конструкторско-технологическому проектированию, синтезу тестов и окончательной верификации принятых проектных решений.

Задачи конструкторского проектирования, как правило, характеризуются большой вычислительной сложностью, обусловленной необходимостью перебора огромного числа различных вариантов решений, причем для получения точного решения требуется выполнить полный перебор, что не представляется возможным. Поэтому актуальной является задача разработки новых и модификации существующих алгоритмов решения задач конструкторского проектирования. К числу таких задач относятся задачи разбиения, размещения, трассировки и др.

Верхний иерархический уровень называют системным или архитектурным. На системном уровне формулируют требования к функциональным и схемным характеристикам, определяют общую архитектуру построения СБИС, выделяют операционные и управляющие блоки. Составляют расписание операций заданного алгоритма, т. е. распределяют операции по временным тактам (scheduling) и функциональным блокам. Тем самым принимают решения по распараллеливанию и/или конвейеризации операций, реализуемым в СБИС.

Задача распределения проектных операций по времени может быть интерпретирована как задача составления временного графика, когда набор из n операций должен быть выполнен на m рабочих местах в течение определенного ограниченного промежутка времени. Каждая операция требует определенного фиксированного времени на определенном рабочем месте.

Задача составления временного графика привлекает внимание исследователей достаточно давно. Однако она является актуальной и в настоящее время. Это объясняется тем, что при планировании необходимо учесть большое число различных факторов, многие из которых не всегда могут быть описаны с помощью классической де-

терминированной теории расписаний. К таким факторам относится, прежде всего, так называемый человеческий фактор.

Время выполнения проектных операций точно неизвестно из-за итерационного характера процесса проектирования, возможных возвратов, а также субъективных факторов. Следовательно, время завершения каждой операции является неопределенным. Кроме того, не всегда возможно создать график, в котором все операции завершаются в срок. Математическая модель, которая используется при решении такой задачи, должна позволить лицу, принимающему решения (ЛПР), выразить свою оценку относительно опоздания каждого задания. Нечеткие множества при этом используются для того, чтобы моделировать неопределенное время выполнения заданий и приоритеты ЛПР по отношению к возможному опозданию завершения каждой операции.

Нечеткое множество определяется функцией принадлежности $\mu_{\widetilde{A}}(x)$, которая назначает каждому объекту x в области исследования X значение, представляющее его степень принадлежности этому нечеткому множеству:

$$\mu_{\widetilde{A}}(x) : \rightarrow [0, 1].$$

Также при решении задачи необходимо учитывать ряд ограничений:

1) последовательность выполнения должна соответствовать определенному порядку;

2) на каждом рабочем месте можно выполнять только одно задание одновременно, и процесс его выполнения не может быть прерван.

Любое решение, удовлетворяющее всем перечисленным ограничениям, будет считаться допустимым графиком.

Функция принадлежности может принимать различные формы: треугольную, трапецеидальную, в виде колокола или s-образную. Выбор формы, как правило, субъективен и позволяет ЛПР выразить свое предпочтение.

Неопределенное время выполнения $\widetilde{p}_{ij}$ моделируется треугольной функцией принадлежности, представленной триплетом $(p_{ij}^1,\ p_{ij}^2,\ p_{ij}^3)$, где p_{ij}^1 и p_{ij}^3 — нижняя и верхняя границы, в то время как p_{ij}^2 — так называемая модальная точка. Пример нечеткого времени выполнения показан на рис. 5.45. Трапецеидальное нечеткое множество используется, чтобы моделировать срок $\widetilde{d}_j$ выполнения каждой операции (рис. 5.46; функция принадлежности представлена в этом случае двойкой $(d_j^1,\ d_j^2)$, где d_j^1 — четкий срок завершения операции, а верхняя граница d_j^2 показывает максимально допустимый срок опоздания, т. е. не должна превышать, например 10 % от значения d_j^1).

Рассмотрим процесс проектирования как совокупность различных операций, причем операции могут выполняться как в автоматизированном, так и в автоматическом режиме. В этом случае будем учитывать n — число процессов $J_1,\ \ldots,\ J_n$ для проекта с заданным временем выпуска $r_1,\ \ldots,\ r_n$ и сроками $d_1,\ \ldots,\ d_n$. Каждое число процессов J_j, $j = 1,\ \ldots,\ n$, состоит из цепочки операций, определенных схемой про-

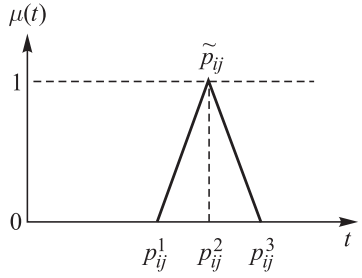


Рис. 5.45. Нечеткое время выполнения операции

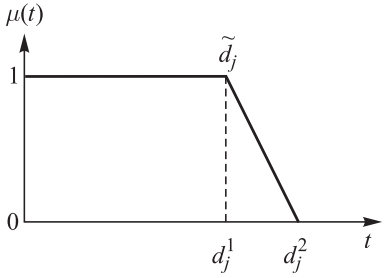


Рис. 5.46. Нечеткий срок опоздания операции

цесса, которая определяется заданными ограничениями, наложенными на операцию. Каждая операция представлена упорядоченной парой (i, j), $i = 1, \ldots, m$, и ее время выполнения обозначается p_{ij}.

Каждая хромосома состоит из двух подхромосом длины m, называемых подхромосомой процессов и подхромосомой операций, используемых для задания последовательности выполнения процессов.

Находим последовательность выполнения операций на каждом задании при выполнении следующих условий:

- минимизировать среднее опоздание C_{AT}:

$$C_{AT} = \frac{1}{n}\sum_{j=1}^{n} T_j; \quad T_j = \max\{0, C_j - d_j\}; \quad j = 1, \ldots, n,$$

  и C_j — время завершения операции J_j на последнем процессе, на котором заканчивается выполнение проекта;
- минимизировать число операций, завершающихся с опозданием, C_{NT}:

$$C_{NT} = \sum_{j=1}^{n} u_j; \quad u_j = 1, \quad \text{если} \quad T_j > 0,$$

  в противном случае $u_j = 0$.

Результирующий график зависит от следующих факторов:

- от использования приоритетов, которые сохраняют заданный порядок выполнения операций внутри каждого процесса;
- от среднего опоздания C_{AT};
- от числа операций, завершающихся с опозданием C_{NT}.

Любое решение, удовлетворяющее всем выше перечисленным ограничениям, называется выполненным графиком.

На рис. 5.47 показана структурная схема разрабатываемого генетического алгоритма.

Алгоритм имеет следующие ограничения:

- на входе задается число процессов и операций в них;

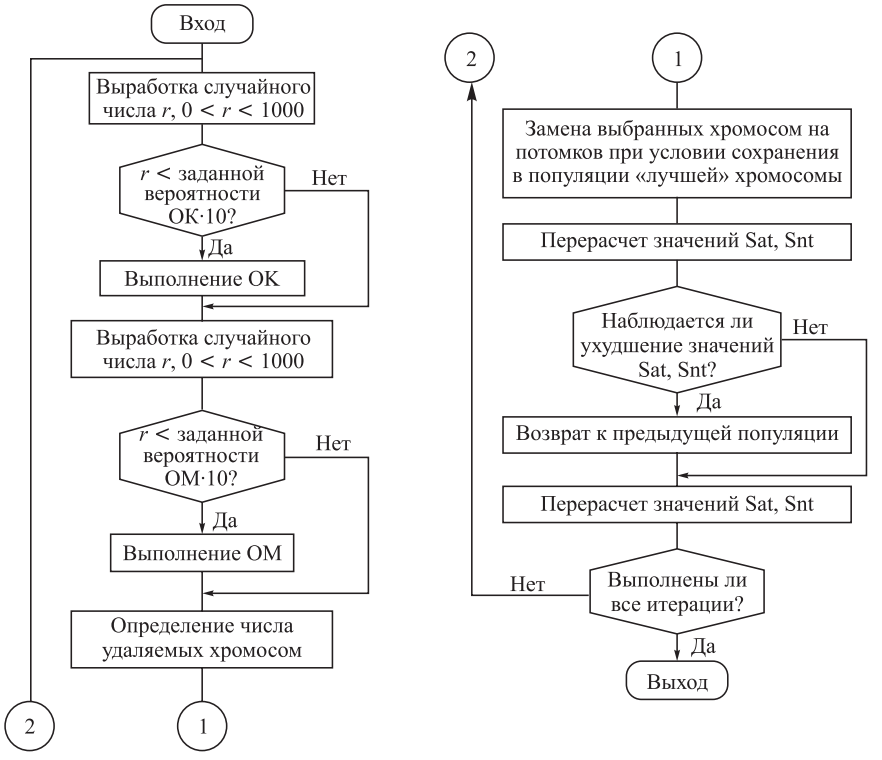


Рис. 5.47. Структурная схема алгоритма

- если значение вероятности не меняется (по умолчанию оно равно 0,8), то пункт «выполнение оператора кроссинговера» пропускается;
- если значение вероятности выполнения оператора мутации не меняется (по умолчанию 0,2), то переход к пункту «определение числа удаляемых хромосом»;
- производится перерасчет значений параметров среднего опоздания и числа поздних операций. При этом если происходит улучшение значений параметров, то переход к п. «Возврат к предыдущей популяции», а если нет, то переход к п. «Перерасчет значений Sat, Snt»;
- критерием останова выполнения алгоритма является пройденное число итераций.

3. *Нечеткий генетический алгоритм для решения задачи трассировки.* Трассировка соединений является, как правило, заключительным этапом конструкторского проектирования ЭВА и состоит в определении линий, соединяющих эквипотенциальные контакты элементов и компонентов, составляющих проектируемое устройство.

В процессе трассировки современных сверх и ультрабольших интегральных схем принято выделять следующие два этапа: глобальная (назначение цепей в определенные регионы) и детальная (прокладка соединений внутри областей) трассировки. В свою очередь детальная трассировка разделяется на трассировку коммутационных блоков и трассировку каналов.

Задача канальной трассировки состоит в проведении соединений между заданными контактами при *минимизации общей площади канала*. Решением задачи канальной трассировки является множество горизонтальных и вертикальных фрагментов каждой цепи.

В генетических алгоритмах (ГА) управляющими параметрами, как правило, являются значения вероятности выполнения генетических операторов кроссинговера (P_c) и мутации (P_m), а также размер популяции. В нечетком генетическом алгоритме происходит динамическое изменение этих параметров при помощи нечеткого логического контроллера (НЛК). Нечеткий логический контроллер на основе входных параметров рассчитывает управляющее воздействие, регулирующее параметры ГА. При большом значении P_c велика вероятность того, что лучшее решение будет «уничтожено» в ходе эволюции при низком же значении сложнее получить лучшее решение, но не гарантируется быстрая сходимость. Высокое значение P_m приводит к слишком большому разнообразию решений и время поиска решения увеличивается, при низком значении можно пропустить некоторые квазиоптимальные решения.

В качестве ограничений задачи используем следующие правила.

1. В каждом гене g_{ij} хромосомы h_{st} может содержаться не более одного значения цепи.

2. Решение (хромосома) h_{st} является допустимым, если канал содержит в себе все размещенные цепи проводников с учетом налагаемых ограничений. Такая хромосома будет включена в множество допустимых решений S.

Целевая функция Q вычисляется в зависимости от количества межслойных переходов и ширины канала:

$$Q = k1 * Q1 + k2 * Q2,$$

где $k1$, $k2$ — коэффициенты локальных критериев, с их помощью можно регулировать влияние того или иного критерия на качество решения. $Q1$ — функция определения количества межслойных переходов, $Q2$ — функция определения ширины канала.

Оптимизация задачи сводится к минимизации критерия Q, т. е. $Q(X) \to \min Q(h_{\text{опт}}) = \min Q(h_{ij})$, где $h_{ij} \subset S$.

Одним из ключевых моментов построения любого генетического алгоритма является кодирование. Сущность кодирования заключается в преобразовании любого числового значения в некоторую последова-

тельность символов конечного алфавита, состоящего обычно из небольшого числа элементов.

Хромосома в данном алгоритме моделируется двумя матрицами: $R1$ и $R2$ (матрица $R1$ для нижнего и матрица $R2$ для верхнего слоев соответственно).

На рис. 5.48 представлена модель хромосомы для двух слоев канала СБИС. Значения g_{ij} и d_{ij} в матрицах $R1$, $R2$ — гены хромосомы, $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$. Число m соответствует количеству магистралей в данном канале, n — количеству выводов в канале,

$$R1 = \begin{bmatrix} g_{11} & g_{21} & g_{31} & \ldots & g_{n1} \\ g_{12} & g_{22} & g_{32} & \ldots & g_{n2} \\ g_{13} & g_{23} & g_{33} & \ldots & g_{n3} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ g_{1m} & g_{2m} & g_{3m} & \ldots & g_{nm} \end{bmatrix}, \quad R2 = \begin{bmatrix} d_{11} & d_{21} & d_{31} & \ldots & d_{n1} \\ d_{12} & d_{22} & d_{32} & \ldots & d_{n2} \\ d_{13} & d_{23} & d_{33} & \ldots & d_{n3} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ d_{1m} & d_{2m} & d_{3m} & \ldots & d_{nm} \end{bmatrix}$$

Рис. 5.48. Структура хромосомы

Возможны три значения гена в хромосоме:

1) $g_{ij} > 0$ — в данном прямоугольнике проходит цепь с номером g_{ij};
2) $g_{ij} = 0$ — данный прямоугольник пустой;
3) $g_{ij} < 0$ — в данном прямоугольнике находится межслойный переход, цепи с номером $|g_{ij}|$.

В матрице $R1$ первая строка соответствует верхним контактам канала, последняя строка — нижнему набору контактов.

Нечеткий генетический алгоритм создает в каждой популяции начальный набор решений. Этот набор состоит из решений, созданных с помощью модифицированного волнового алгоритма. Работа алгоритма заключается в повторении этапов решения задачи. В течение одной итерации каждый процессор выполняет последовательный генетический алгоритм с его популяцией. В конце работы лучшая хромосома, которая существует (или существовала), составляет конечное решение задачи.

После формирования начальной популяции происходит последовательное выполнение генетических операторов. В алгоритме применяются модифицированные генетические операторы, ориентированные на специфику задачи, после чего вступает в работу нечеткий логический контроллер, который, используя входные данные, производит корректировку значений вероятностей кроссинговера и мутации.

Из всех полученных в ходе работы алгоритма решений выбирается лучшее и запоминается. Работа алгоритма заканчивается по достижению последней итерации. Схема генетического алгоритма канальной трассировки приведена на рис. 5.49. Временная сложность предлагаемого алгоритма в среднем составляет $O(N^2)$–$O(N^3)$.

4. *Нечеткий генетический алгоритм решения задачи коммивояжера.* Постановку задачи о коммивояжере можно описать следующим

Рис. 5.49. Структурная схема нечеткого генетического алгоритма

образом: Есть некий коммивояжер, которому необходимо посетить N городов, не заезжая в один и тот же город дважды, и вернуться в исходный пункт. При этом длина (стоимость) пройденного пути должна быть минимальной. С точки зрения теории графов задача выглядит так: дан граф $G = (X, U)$, где $|X| = n$ — множество вершин (городов), $|U| = m$ — множество ребер (путей) между городами; $D_{n\times n}$ — матрица расстояний (стоимостей), где d_{ij}, $i, j \in \{1, 2, \ldots, n\}$, — это длина пути (стоимость переезда) из города (вершины) x_i в x_j.

Для решения задачи коммивояжера предлагается использовать нечеткий генетический алгоритм. При этом мы используем следующие настройки ГА:

1. Случайное формирование начальной популяции, например, на основе метода «дробовика».
2. Кодирование решений в виде последовательности проходимых путей (ребер).
3. Оператор кроссинговера на основе жадной эвристики.

4. Оператор мутации.
5. Оператор отбора на основе «колеса рулетки».

Основным моментом является использование НЛК, использующего в качестве входных параметров среднее и лучшее значение ЦФ.

Нечеткий логический контроллер оперирует лингвистическими переменными, такими, как NL (negative large) — «значительно хуже», NS (negative small) — «незначительно хуже», ZE (zero) — «частичное изменение», PS (positive small) — «незначительно лучше», PL (positive large) — «значительно лучше».

Формирование управляющего воздействия НЛК осуществляет на основе оценки двух параметров:

$$e_1(t) = \frac{f_{\text{ave}}(t) - f_{\text{best}}(t)}{f_{\text{ave}}(t)},$$

$$e_2(t) = \frac{f_{\text{ave}}(t) - f_{\text{ave}}(t-1)}{f_{\text{best}}(t)},$$

где t — временной шаг; $f_{\max}(t)$ — лучшее значение ЦФ на итерации t; $f_{\text{ave}}(t)$ — среднее значение ЦФ на итерации t; $f_{\text{ave}}(t-1)$ — среднее значение ЦФ на итерации $(t-1)$.

Параметры e_1, e_2 заданы на следующих интервалах:

- $e_1 \in [0; 1]$;
- $e_2 \in [-1; 1]$.

Выходной параметр $\Delta \Pr_c(t)$ определяющий вероятность выполнения кроссинговера задан на интервале: $\Delta \Pr_c(t) \in [-0{,}1; 0{,}1]$. Выходной параметр для оператора мутации $\Delta \Pr_m(t)$ тот, же что и для кроссинговера. Отсюда видно, что вероятности могут меняться не более, чем на 10 %. Тогда используем четкие значения, чтобы изменить вероятности $\Pr_c$ и $\Pr_m$ следующим образом:

$$\Pr_c(t) = \Pr_c(t-1) + \Delta \Pr_c(t),$$

$$\Pr_m(t) = \Pr_m(t-1) + \Delta \Pr_m(t).$$

Следовательно, НЛК представляет собой двумерный набор нечетких правил:

Таблица 5.1. Нечеткие правила для ОК ($\Delta \Pr_c(t)$)

	e_2				
e_1	**NL**	**NS**	**ZE**	**PS**	**PL**
PL	NS	ZE	PS	PS	PL
PS	ZE	ZE	ZE	ZE	ZE
ZE	NS	NL	ZE	NL	NS

Таблица 5.2. Нечеткие правила для ОМ ($\Delta \Pr_m(t)$)

	e_2				
e_1	**NL**	**NS**	**ZE**	**PS**	**PL**
PL	PS	ZE	PS	NS	NL
PS	ZE	ZE	NS	ZE	NS
ZE	PS	PL	ZE	PL	PS

Так, например, если на входе нечеткого логического контроллера заданы следующие параметры $e_1 = \mathrm{ZE}$ и $e_2 = \mathrm{NL}$, то согласно приведенным таблицам получим

$$\Delta \Pr_c(t) = \mathrm{NS}, \qquad \Delta \Pr_m(t) = \mathrm{PS}.$$

Следовательно, значения вероятностей кроссинговера и мутации определяются выражениями:

$$\Pr_c(t) = \Pr_c(t-1) + \mathrm{NS},$$
$$\Pr_m(t) = \Pr_m(t-1) + \mathrm{PS}.$$

Итак, при заданных значениях параметров e_1 и e_2 вероятность кроссинговера «немного» уменьшится, а вероятность мутации «значительно» увеличится.

Очевидно, что для эффективной работы нечеткого генетического алгоритма необходимо провести тестирование и настройку как входных сигналов, так и соответствующих им управляющих воздействий нечеткого логического контроллера.

Структура предложенного нечеткого генетического алгоритма выглядит следующим образом (рис. 5.50):

1 Начало НГА
2 $t = 0$ Iteration counter
3 `Формирование начальной популяции` $P(t)$
4 `Расчет ЦФ` популяции $P(t)$
5 `Пока` (не выполнено условие останова)
6 {
7 $t = t + l$
8 `Селекция`
9 `ОК` $P(t)$
10 `ОМ` $P(t)$
11 `Расчет ЦФ` $P(t)$
12 `Отбор` $P(t)$
13 Регулировка параметров ГА

```
14 {
15 нлк (e1, e2)
16 Обновляем параметры
17 }
18 }
19 Конец НГА
```

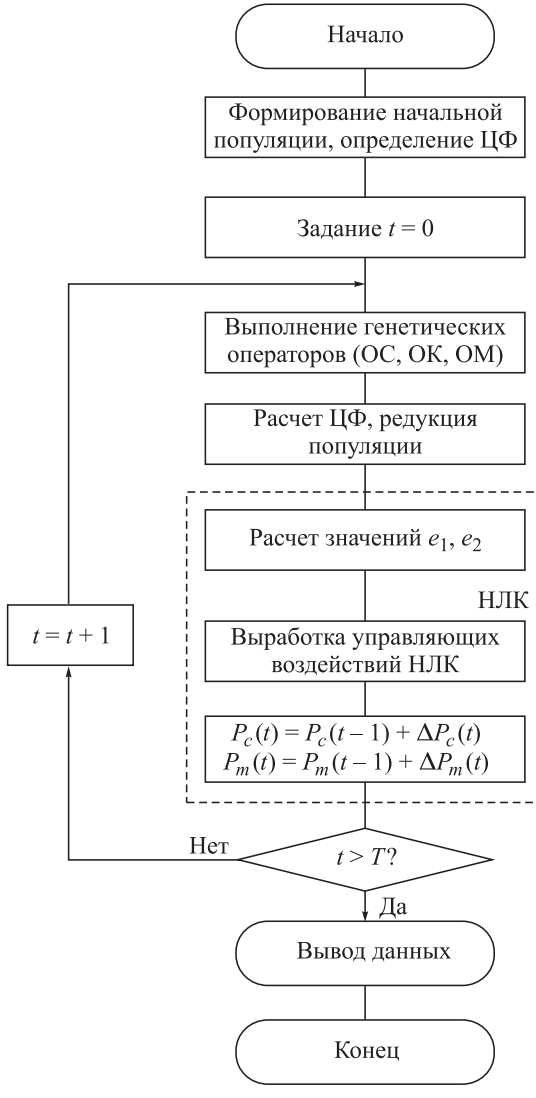


Рис. 5.50. Структурная схема НГА

5.9. Выводы

В данной главе приведен ряд примеров, иллюстрирующих эффективность применения генетических алгоритмов для решения различных практических оптимизационных задач. Очевидно, что эффективность использования генетических алгоритмов зависит как от правильной настройки параметров алгоритма, так и от размерности решаемой задачи. Так, для решения простых задач небольшой размерности наиболее эффективными будут эвристические итерационные алгоритмы, тогда как при решении сложных NP-полных задач генетические алгоритмы доказали свою эффективность.

Класс решаемых с помощью генетических алгоритмов задач на самом деле гораздо шире представленных в данной главе примеров. На сегодняшний день генетические алгоритмы с успехом применяются для решения различных задач проектирования цифровой и аналоговой техники, задач построения интеллектуальных систем, для решения задач оптимизации и прогнозирования и т. д. Однако ограниченный объем не позволяет перечислить все возможные направления использования.

В заключение хотелось бы еще раз подчеркнуть, что генетические алгоритмы являются на сегодняший день перспективным и прогрессирующим направлением научного прогресса, однозначно доказавшим свою эффективность и право на существование.

5.10. Контрольные вопросы

1. Какие оптимизационные задачи эффективно решать при помощи генетических алгоритмов?
2. Какие задачи называются NP-полными?
3. В чем основная идея применения генетического алгоритма для решения задачи разбиения графа на части?
4. В чем основная идея применения генетического алгоритма для решения задачи коммивояжера?
5. В чем основная идея применения генетического алгоритма для решения задачи раскраски графа?
6. В чем основная идея применения генетического алгоритма для решения задачи выделения клик в графе?
7. В чем основная идея применения генетического алгоритма для решения задачи выделения независимых подмножеств в графе?
8. В чем основная идея применения генетического алгоритма для решения задачи определения планарности графа?
9. В чем основная идея применения генетического алгоритма для решения задачи определения изоморфизма графов?
10. Каким образом можно применить генетический алгоритм для решения задачи изоморфного вложения?
11. Каким образом можно сравнить эффективность генетического и эвристического алгоритмов решения оптимизационных задач?

12. К какому классу алгоритмов относятся генетические алгоритмы?
13. Что является критерием эффективности генетического алгоритма?
14. Какова временная сложность генетического алгоритма для решения задачи разбиения графа на части?
15. Какова временная сложность генетического алгоритма для решения задачи коммивояжера?
16. Какова временная сложность генетического алгоритма для решения задачи раскраски графа?
17. Какова временная сложность генетического алгоритма для решения задачи выделения клик в графе?
18. Какова временная сложность генетического алгоритма для решения задачи выделения независимых подмножеств в графе?
19. Какова временная сложность генетического алгоритма для решения задачи определения планарности графа?
20. Какова временная сложность генетического алгоритма для решения задачи определения изоморфизма графов?
21. В чем заключается эвристика определения паросочетания в двудольном графе?
22. Каким образом выполняется склеивание диагонали для нахождения максиамального паросочетания в двудольном графе?

5.11. Упражнения

1. Запишите стуктуру генетического алгоритма для решения задачи нахождения гамильтонова цикла в графе.
2. Опишите структуру генетического алгоритма для решения задачи разбиения графа на части.
3. Опишите структуру генетического алгоритма для решения задачи коммивояжера.
4. Опишите структуру генетического алгоритма для решения задачи раскраски графа.
5. Опишите структуру генетического алгоритма для решения задачи выделения клик в графе.
6. Опишите структуру генетического алгоритма для решения задачи выделения семейства клик в графе.
7. Опишите структуру генетического алгоритма для решения задачи выделения максимальной клики в графе.
8. Опишите структуру генетического алгоритма для решения задачи выделения минимальной клики в графе.
9. Опишите структуру генетического алгоритма для решения задачи выделения семейства независимых подмножеств в графе на основе жадных стратегий.
10. Опишите структуру генетического алгоритма для решения задачи выделения независимых подмножеств в графе.
11. Постройте схему генетического алгоритма для определения максимального и минимального независимого подмножества в графе.

12. Опишите структуру генетического алгоритма для определения планарности графа.
13. Постройте схему генетического алгоритма определения планарности графа на основе критерия Понтрягина–Куратовского.
14. Постройте схему генетического алгоритма определения планарности графа на основе критерия Уитни.
15. Постройте схему генетического алгоритма определения планарности графа на основе критерия Харари–Татта.
16. Постройте схему генетического алгоритма определения планарности графа на основе критерия Мак Лейна.
17. Опишите структуру генетического алгоритма для построения плоской укладки графа.
18. Опишите структуру генетического алгоритма для определения изоморфизма неоднородных графов.
19. Опишите структуру генетического алгоритма для определения изоморфизма однородных графов.
20. Постройте схему генетического алгоритма для определения изоморфного вложения графов.
21. Постройте псевдокод алгоритма определения максимального паросочетания в двудольном графе.
22. Постройте структурную схему алгоритма определения максимального паросочетания в неориентированном графе
23. Опишите структуру генетического алгоритма для решения переборных комбинаторно-логических задач на графах.

Глоссарий к разделу 5

Временная сложность алгоритма (ВСА) — зависимость времени работы алгоритма от числа входов (исходных данных задачи).

Полиномиальные алгоритмы — алгоритмы, временная сложность которых составляет $O(n), O(n^2), O(n^3), \ldots$.

Экспоненциальные алгоритмы — алгоритмы, временная сложность которых составляет $O(n!)$ и т. д., $O(n^n), O(n^{2n}), O(n^{3n}), \ldots$.

NP-полные задачи — оптимизационные задачи, для которых не найдены полиномиальные алгоритмы, однако и не доказано, что их не существует.

Разбиение графа на части заключается в минимизации критерия K (суммарное число связей между частями разбиения).

Кластер — часть графа. Вершины кластера соединены внутренними ребрами с остальными вершинами графа. Внешние ребра графа не входят в подмножество ребер кластера.

Раскраска вершин графа — разбиение множества вершин на непересекающиеся классы (подмножества) такие, что внутри каждого подмножества не должно содержаться смежных вершин. Если каждому подмножеству поставить в соответствие определенный цвет, то верши-

ны внутри этого подмножества можно окрасить в один цвет, а вершины другого подмножества — в другой цвет, и так далее до раскраски всех подмножеств.

Внутренне устойчивое* (*независимое*) *подмножество вершин графа — подмножество вершин графа, в котором любые две вершины не смежны.

Максимальное внутренне устойчивое подмножество вершин графа — подмножество вершин, добавление к которому любой вершины графа делает его не внутренне устойчивым.

Клика — подмножество из максимального числа смежных между собой вершин в графе.

Плоский граф — граф, для которого существует такое изображение на плоскости его вершин и ребер, что каждая вершина изображается на плоскости отдельной точкой. При этом ребра простой кривой, имеющей концевые точки, пересекаются только в общих точках, т. е. в вершинах.

Планарный граф — граф, для которого существует возможность получения плоской укладки.

Грань — область плоскости, ограниченная ребрами плоского графа и не содержащая внутри себя ни вершин, ни ребер.

Задача распознавания изоморфизма двух графов — задача определения: существует ли взаимно-однозначное отображение, переводящее один граф в другой, т. е. $\varphi : X_1 \to X_2$ такое, что $U = (x, y) \in U_1$ тогда и только тогда, когда $(\varphi(x), \varphi(y)) \in U_2$.

Паросочетанием называется подмножество ребер, не имеющих общих концов, причем каждое ребро графа смежно одному ребру из данного подмножества.

Максимальное паросочетание — паросочетание, содержащее максимально возможное число ребер.

Список литературы к разделу 5

1. *Андерсон Дж.* Дискретная математика и комбинаторика. — М.: Изд. дом «Вильямс», 2003.
2. *Батищев Д. А.* Генетические алгоритмы решения экстремальных задач. — Воронеж: Изд-во ВГТУ, 1995.
3. *Берштейн Л. С.*, *Боженюк А. В.* Нечеткие графы и гиперграфы. — М.: Научный мир, 2005.
4. *Гладков Л. А.*, *Курейчик В. В.*, *Курейчик В. М.*, *Сороколетов П. В.* Биоинспирированные методы в оптимизации. — М.: ФИЗМАТЛИТ, 2009.
5. *Джонс М. Т.* Программирование искусственного интеллекта в приложениях. — М.: ДМК Пресс, 2004.
6. *Емельянов В. В.*, *Курейчик В. В.*, *Курейчик В. М.* Теория и практика эволюционного моделирования. — М.: Физматлит, 2003.
7. *Комарцова Л. Г.*, *Максимов А. В.* Нейрокомпьютеры. — М.: Изд-во МГТУ, 2002.

8. *Кормен Т.*, *Лейзерсон И.*, *Ривест Р.* Алгоритмы: построения и анализ. — М.: МЦНМО, 2000.
9. *Кристофидес Н.* Теория графов. Алгоритмический подход. — М.: Мир, 1978.
10. *Кроновер Р. М.* Фракталы и хаос в динамических системах. Основы теории. — М.: Постмаркет, 2000.
11. *Курейчик В. В.* Эволюционные методы решения оптимизационных задач. — Таганрог: Изд-во ТРТУ, 1999.
12. *Курейчик В. М.* Дискретная математика. Ч. 3. Оптимизационные задачи на графах. — Таганрог: Изд-во ТРТУ, 1998.
13. *Курейчик В. М.* Генетические алгоритмы и их применение. — Таганрог: Изд-во ТРТУ, 2002.
14. *Курицкий Б. Я.* Оптимизация вокруг нас. — Л.: Машиностроение, 1989.
15. *Макконелл Дж.* Анализ алгоритмов. Вводный курс. — М.: Техносфера, 2002.
16. *Новиков Ф. А.* Дискретная математика для программистов. — С.-Пб.: Питер, 2000.
17. *Таха*, *Хэмди А.* Введение в исследование операций. — М.: Изд. дом «Вильямс», 2001.
18. *Kureichik V.M., Malioukov S.P., Kureichik V.V., Malioukov A.S.* Genetic Algorithms for Applied CAD Problems. — Berlin, Heidelberg: Springer-Verlag, 2009.

Графы соединяют в себе геометрическую наглядность с математической содержательностью и с возможностью обходиться без громоздкого аппарата. Они являются универсальным инструментом представления различных задач оптимизации.

Примеры использования генетических алгоритмов для решения сложных комбинаторно-логических задач переборного типа на графах демонстрируют эффективность данного метода

Заключение

Природа не знает никаких прав,
ей известны только законы.

Д. Адамс

Генетические алгоритмы — новая фундаментальная область исследований, основанная Д. Холландом, впервые примененная для решения задач оптимизации и распознавания образов. Генетические алгоритмы моделируют в определенной степени механизмы развития органического мира на Земле. В настоящее время генетические алгоритмы — это мощный адаптивный поисковый метод, основанный на эволюционных факторах конструирования популяции, естественного отбора, селекции, мутации, эволюционной теории Ч. Дарвина.

В технике наибольшее применение находят простые и объектно-ориентированные генетические алгоритмы, выполняющие 4 основных функции: селекция, кроссинговер, мутация и рекомбинация. Основной проблемой применения таких генетических алгоритмов при решении задач оптимизации является предварительная сходимость, т. е. попадание в локальные оптимумы, выход из которых затруднен. В учебном пособии представлены модифицированные схемы генетического поиска, позволяющие в большинстве случаев решать проблемы предварительной сходимости алгоритмов.

Предложенные схемы эффективно используются для решения комбинаторно-логических задач на графах. Генетические алгоритмы позволяют одновременно анализировать некоторое подмножество решений, формируя квазиоптимальные решения. Временная сложность алгоритмов зависит от параметров генетического поиска и числа генераций.

Основные проблемы при реализации генетических алгоритмов заключаются в следующем. Важно найти оптимальный размер популяции, достаточное число генераций. Наибольшие затруднения вызывают вопросы управления процессом генетического поиска. При построении некоторого подмножества популяций появляется возможность эффективного использования параллельных вычислений с использованием транспьютеров, гиперкубовых ЭВМ, многопроцессорных вычислительных систем, нейронных компьютеров и др.

Отметим, что генетические алгоритмы — мощная стратегия выхода из локальных оптимумов. Она заключается в параллельной обработке множества альтернативных решений с концентрацией поиска на наиболее перспективных из них, причем периодически в каждой итерации

можно проводить стохастические изменения в менее перспективных решениях.

Решение рассмотренных в учебном пособии прикладных задач с использованием разработанных генетических алгоритмов показало актуальность использования методов моделирования эволюции и подтвердило важность расширения исследований в этой области.

Для лучшего усвоения материала авторами была использована современная методика обучения на основе «решебников». В начале каждой главы приводится краткое изложение теории, затем подробно рассматриваются примеры. Кроме того, приводятся контрольные задачи, упражнения и глоссарий с пояснением основных терминов. Задачи и упражнения составлены авторами на основе списка литературы, приведенного в конце пособия. Опыт преподавания в вузах России, США, Германии и Японии показал перспективность такого метода представления материала.

Основное влияние теория генетических алгоритмов оказывает на развитие вычислительной техники, микроэлектроники, нанотехнологий, физики и других наук. В настоящее время применение генетических алгоритмов является повсеместным во всех областях науки и техники, поэтому они являются не только фундаментом современной математики, но и основным звеном математического образования.

Теоретический материал состоит из большого числа моделей, аксиом, теорем и формулировок, используемых в пособии понятий. Авторы стремились показать широкие возможности применения теории генетических алгоритмов, ее универсальность и специализированность для решения задач информатики и вычислительной техники.

Теория генетических алгоритмов постоянно развивается и совершенствуется. Данное учебное пособие является одним из первых в нашей стране. Авторы надеются, что данное пособие явится толчком к дальнейшему развитию теории и практики генетических алгоритмов в нашей стране.

ПРИЛОЖЕНИЯ

Приложение 1. Элементарные сведения из теории алгоритмов

1.1. Понятие алгоритма. Однозначного и точного определения алгоритма не существует. В связи с этим приведем несколько наиболее распространенных определений понятия «алгоритм».

Алгоритм (*algorithm*) — формально описанная вычислительная процедура, получающая исходные данные, называемые также входом алгоритма или его аргументом и выдающая результат вычисления на выход.

Алгоритм — набор правил, указывающих определенные действия, в результате которых входные данные преобразуются в выходные. Последовательность действий в алгоритме называется *алгоритмическим процессом*, а каждое отдельное действие — *шагом алгоритма*.

Понятие *алгоритма* также можно интуитивно определить как конечную последовательность точно заданных правил решения произвольного класса задач.

Интуиция — знание, полученное на основе опыта, но не подвергнутое научному анализу.

К алгоритмам не относятся запрещающие и разрешающие правила. Например: «Не курить! Не сорить!»

Особенность алгоритмов — дискретный характер процесса, определяемого самим алгоритмом. Алгоритмы строятся для решения тех или иных вычислительных задач. Формулировка задачи описывает, каким требованиям должно удовлетворять решение задачи, а алгоритм, решающий эту задачу, находит объект, этим требованиям удовлетворяющий.

Алгоритм имеет дело с тремя типами данных:

1) входные данные;

2) промежуточные данные;

3) выходные данные.

Алгоритмы, в которых решения поставленных задач сводятся к арифметическим действиям, называется *численными алгоритмами*.

Обычно алгоритмы представляют в виде устройства, называемого «черным ящиком».

Для размещения данных алгоритма требуется память.

В теоретическом плане считают, что память состоит из одинаковых ячеек, причем каждая ячейка памяти может содержать один или несколько символов данных.

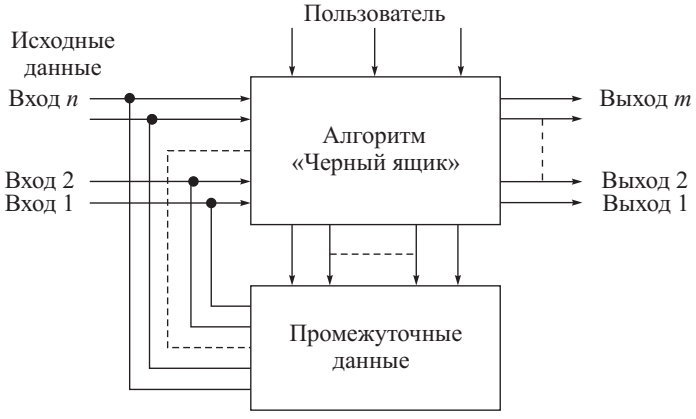


Рис. П.1

В существующих в настоящее время алгоритмах последовательность выполняемых элементарных шагов детерминирована, т. е. после каждого шага алгоритма указывается номер следующего шага. От каждого алгоритма требуется остановка после конечного числа шагов, дающих требуемый результат. Различают описание алгоритмов и процесс их реализации.

Алгоритм считают правильным, если на любом допустимом (для данной задачи) входе он заканчивает работу и выдает результат, удовлетворяющий требованиям задачи. В этом случае говорят, что алгоритм решает данную вычислительную задачу. Неправильный алгоритм может (для некоторого входа) вовсе не остановиться или дать неправильный результат.

Алгоритм может быть записан на русском или английском языке, в виде компьютерной программы или даже в машинных кодах, важно только, чтобы процедура вычислений была четко описана. Алгоритмы также могут записываться с помощью псевдокода, представляющего симбиоз из основных операторов различных языков программирования (`Си`, `Паскаль`, `Алгол`). Разница в том, что в псевдокоде могут использоваться не только эти операторы, но и словесное описание действий алгоритма, а также отсутствуют технологические подробности (например, процедура обработки ошибок), имеющиеся в реальной программе.

Каждый алгоритм обычно связан с двумя алфавитами (или их называют языками). На одном языке алгоритм сформулирован сам, а предложения другого является допустимыми для него вариантами исходных данных.

Первый язык называется алгоритмическим языком. Второй называется языком операндов.

Операнды — это объекты, над которыми выполняются операции, предписываемые алгоритмами.

Существуют два вида определений в алгоритмах. Первый называется «прямые определения». В них новые понятия выражаются через одно или несколько известных. Второй вид называется «порочный круг». Это определения, в которых новое понятие выводится либо из самого себя, либо из другого понятия, которое было из него выведено.

Часто используют рекурсивные определения. Они состоят из двух частей: 1-я — циклическая, 2-я — прямая часть определения. Она является входом в циклическую.

1.2. Классификация алгоритмов. Существует большое число различных классификаций алгоритмов. Рассмотрим некоторые из них.

В отдельные классы выделяют имитирующие и эмпирические алгоритмы.

Имитирующие алгоритмы создаются на основе описаний действий объектов, наблюдений, зависимости исходных данных от изменяющихся условий.

Эмпирические алгоритмы разрабатываются на основе наблюдений за каким-то процессом действий.

Эвристические алгоритмы, или эвристики, определяются как алгоритмы:

- находящие обычно «хорошие», но не обязательно наилучшие (оптимальные) решения,
- имеющие более простую реализаию, чем любой известный алгоритм, гарантирующий оптимальное решение.

Понятие «хороший» меняется от задачи к задаче. Универсальной структуры описания эвристических алгоритмов не существует.

Например, во многих задачах информатики требуется выборка A предметов из множества B предметов. Один эвристический подход состоит в том, чтобы выбирать предметы с вероятностью A/B. Здесь каждый предмет принимается или отвергается в момент его просмотра и можно делать только один просмотр вдоль всего набора предметов. В общем виде эта процедура не гарантирует, что будет выбрано в точности A предметов. Другой эвристический подход при выборе A чисел из B состоит в том, чтобы многократно генерировать случайные целые числа до тех пор, пока не окажется A различных.

Необходимо привести еще одну классификацию. Согласно ей все алгоритмы можно разделить на три группы:

1) линейные;
2) разветвляющиеся;
3) циклические.

Алгоритм называется *линейным*, если он предусматривает получение результата путем однократного выполнения одной и той же последовательности действий при каждом значении исходных данных.

Алгоритм называется *разветвляющимся*, если он предусматривает выбор одной из нескольких альтернатив последовательностей действий в зависимости от исходных и промежуточных данных. Каждая из этих последовательностей называется *ветвью* алгоритма. В отличие от линейного, разветвляющий алгоритм имеет, по крайней мере, один логический блок.

Наиболее сложным является *циклический* алгоритм (по структуре), выполняемый путем многократного повторения некоторой последовательности действий. По способу определения числа повторений цикла выделяют следующие алгоритмы:

- последовательные с явно заданным числом повторений цикла;
- итерационные имеют одинаковую структуру с последовательными, но выход из цикла осуществляется не после того, как цикл повторится некоторое число раз, а при выполнении общего условия, связанного с проверкой значения монотонности, изменения в алгоритме исходной величины.

1.3. Универсальные алгоритмические модели. Существуют три основных типа *универсальных алгоритмических моделей*:

1. Преобразование слов в произвольных абстрактных алфавитах.
2. Преобразование числовых функций.
3. Представление алгоритмов, как некоторого детерминированного устройства, способного выполнять простейшие операции.

1.4. Преобразование слов в абстрактных алфавитах. В этом типе алгоритмических моделей под алгоритмом понимают задаваемые соответствия между словами в *абстрактных алфавитах*. Под абстрактными алфавитами понимается конечная совокупность объектов, называемых буквами или символами этого алфавита. *Символами* абстрактных алфавитов считают буквы, цифры, любые знаки, слова, предложения, любые тексты. Следовательно, абстрактные алфавиты — это конечное множество различных символов, и, подобно множеству, его можно задать путем перечисления элементов или символов.

Например, A — абстрактный алфавит:

$$A = \{+, a, \text{студент}, b, c, \text{учебник}\}.$$

Абстрактный алфавит может быть многоступенчатым, в этом случае каждый символ может сам являться абстрактным алфавитом или его частью.

Алгоритмами называют преобразование слов в абстрактные алфавиты. Соответственно словом в абстрактных алфавитах называют любую конечную упорядоченную последовательность символов:

$$\text{Слово}_A = \langle +, a\rangle, \ \ \text{Слово}_B = \langle b, +c\rangle, \ \ \text{Слово}_C = \langle\,\rangle, \ \ \text{Слово}_D\langle +\rangle.$$

Число символов в слове абстрактных алфавитов называется *длиной* этого слова. Пустое слово в абстрактные алфавиты обозначается $\varnothing$.

Алфавитным оператором называют функцию, которая задает соответствие между словами входного и выходного абстрактных алфавитов.

Например, некоторая структура с заданным X входным и выходным Y абстрактными алфавитами:

$$X = \{a, b, \text{ЭВМ}, \text{ПЭВМ}\};\ \ Y = \{a + b, a * b, \text{система}, \text{ВЦ}\},$$

является алгоритмом. Тогда нам нужно построить алфавитный оператор, который перерабатывает входные слова алфавита X в выходные слова алфавита Y.

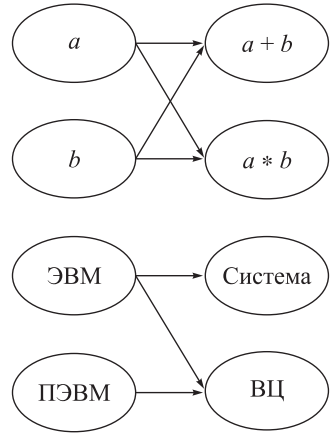


Рис. П.2

Алфавитные операторы бывают однозначные и многозначные. В однозначном алфавитном операторе каждому входному слову ставится в соответствие одно выходное слово. В многозначных алфавитных операторах каждому входному слову может быть поставлено в соответствие некоторое множество выходных слов. Алфавитный оператор, который не ставит в соответствие входному слову никакого выходного, неопределен на этом слове. В рассматриваемом выше примере мы имеем многозначный алфавитный оператор.

Множество слов, на котором алфавитный оператор определен, называется его областью определения. Алфавитные операторы бывают с конечной и бесконечной областями определения. Если область определения алфавитного оператора конечна, то его обычно задают таблицей соответствий. В левой части записывают слова, входящие в область определения, в правую часть таблицы — слова, получающиеся в результате применения алфавитного оператора к входным словам.

Если область определения алфавитного оператора бесконечна, то его задают набором правил, которые за конечное число шагов позволяют найти выходное слово, соответствующее заданному входному.

Алфавитный оператор, задаваемый с помощью конечной системы правил, называется *алгоритмом*.

Следует различать понятие алгоритма и алфавитного оператора.

В алфавитном операторе основное — это соответствие между словами, а не способ его установления. В алгоритме же основное — способ установления соответствия между словами. Следовательно, *алгоритм* — это алфавитный оператор с правилами, определяющими его действие. Отметим, что два алгоритма называются *равными*, если равны их алфавитный оператор и совпадают системы правил, задающих действие этих алгоритмов на входные слова. Алгоритмы считаются

эквивалентными, если они имеют совпадающие алфавитные операторы, но не совпадающие способы их задания.

1.5. Числовые функции. Из определения алгоритма следует, что он ставит в соответствие исходным данным выходные. Поэтому каждому алгоритму можно поставить в соответствие функцию, которую он вычисляет.

Ко второму типу алгоритмических моделей относятся *числовые функции*. Функцию называют вычислимой, если имеется способ получения ее значений. В алгоритмах используются рекурсивные функции, представляющие собой частичный класс вычислимых функций. Процедуру, которая прямо или косвенно обращается к себе, называют *рекурсивной*.

Рекурсия — способ задания функций, когда ее значения для произвольных значений аргументов выражаются известным образом через значения определяемой функции для меньших значений аргументов.

1.6. Сложность алгоритмов. Важнейшая практическая задача — анализ алгоритма. При этом необходимо получение оценок или границ для объема памяти и времени работы, которое потребуется алгоритму для успешной обработки конкретных данных. Необходимо иметь количественный критерий для сравнения двух алгоритмов, созданных для решения одной и той же задачи. Важно установить абсолютный критерий, т. е. указывающий, когда можно считать решение задачи оптимальным.

Один из основных результатов Тьюринга — разделение всех представимых задач в математике на два типа (класса).

1-й класс. Задачи, для которых алгоритмы никогда не могут быть написаны, т. е. задачи постоянно не решаемые.

Пример.

а) задача о квадратуре круга;

б) задача трисекции угла — найти общий алгоритм деления произвольного угла на три равные части.

За счет расширения набора допустимых операций можно иногда часть проблем, для которых нет алгоритма, сделать разрешимыми. В качестве основного средства получения новых материальных объектов должны служить алгоритмы. Это направление в математике называется конструктивным.

2-й класс. Это класс решаемых задач, т. е. по которым могут быть написаны алгоритмы:

а) полиномиальные алгоритмы,

б) экспоненциальные алгоритмы.

Обычно алгоритмы изменяются во времени.

Экспоненциальными алгоритмами называются алгоритмы, имеющие свойства экспоненциального роста. К ним относятся алгоритмы

типа 2^n, $n!$, $n!!$, где n — количество входов алгоритма, т. е. алгоритмы, в которых время решения задачи зависит от величин 2^n, $n!$, $n!!$ и т. д.

Полиномиальные алгоритмы имеют вид $n, n^2, n^3, \ldots, n^n$ — количество входов алгоритма, т. е. алгоритмы, где время решения зависит от $n, n^2, n^3, \ldots$.

Изменение алгоритма во времени или время, затраченное алгоритмом как функция размерности задачи, называется *временной сложностью алгоритма* и записывается ВСА.

ВСА — это количество единиц времени, требуемое для обработки входных данных задачи.

Отметим, что согласно Д. Кормену время работы алгоритма $T(n) = \Theta(n^3)$ означает, что найдутся такие константы C_1, $C_2 > 0$ и такое число n_0, что $C_1 n^3 \leqslant T(n) \leqslant C_2 n^3$ при всех $n \geqslant n_0$. Эта запись включает две оценки: верхнюю и нижнюю. Следовательно, для любых двух функций $f(n)$ и $g(n)$ свойство $f(n) = \Theta(g(n))$ выполнено, когда $f(n) = O(g(n))$. Это выражение является нижней границей, если найдутся такая константа $c > 0$ и n_0, что $0 \leqslant f(n) \leqslant c(g_n)$ для всех $n \geqslant n_0$. Выражение $f(n) = \Omega(g(n))$ — верхняя граница, если найдутся $c > 0$ и n_0, $0 \leqslant cg(n) \leqslant f(n)$ для всех $n \geqslant n_0$. Обозначается ВСА $O[f_A(n)]$. Функция $f_A(n)$ дает верхнюю границу для максимального числа операций, которые должен выполнить алгоритм для решения любой задачи размерности n.

Основные операции — сдвиг, сравнение, сложение и тому подобное. Существует теорема, что для достаточно больших n линейные алгоритмы всегда эффективнее квадратичных, кубичных и т. д. алгоритмов. Четкой границы между экспоненциальными и полиномиальными алгоритмами нет. Все алгоритмы, у которых ВСА $O(n)$, $O(n^2)$, $O(n^3)$, считаются эффективными и применяются при решении практических задач.

Все алгоритмы, у которых ВСА равно $O(n^6)$ и выше, считаются не эффективными для использования на практике.

Кроме ВСА важной характеристикой является сложность алгоритма. *Сложность алгоритма* характеризуется числом выполняемых операций N и общим объемом используемой информации.

Поскольку выполнение разных типов операций требует разного времени, сложность алгоритма зависит не только от общего количества операций, но и от количества типов операций и количества операций каждого типа, используемых рассматриваемым алгоритмом. С числом операций алгоритма связано время его выполнения на конкретной ЭВМ. Объем информации связан с объемом памяти ЭВМ, необходимой для реализации этого алгоритма. Значит, время реализации алгоритма есть функция $T = f(N, V)$.

Но выполнение различных операций требует различного времени. Поэтому для определения сложности алгоритма необходимо знать не только общее количество операций, но и количество используемых

в алгоритме типов операций и количество операций каждого типа в отдельности.

Количество операций для алгоритма получается следующим образом. На основании анализа алгоритма можно получить вектор встречаемости операций в алгоритме:

$$N = \{Q_1, Q_2, \ldots, Q_r\},$$

где Q_i — количество операций в исследуемом классе задач, $N = \sum_{i=1}^{r} Q_i$; $P_i = Q_i/N$ — относительная частота операций.

Умножая время выполнения каждой операции на количество этих операций и суммируя результаты по всем операциям, получим время реализации алгоритма:

$$T = \sum_{i=1}^{r} Q_i t_i.$$

Принимая за эталон операцию сложения, можно определить общее количество приведенных операций. Для этого необходимо знать относительные веса различных операций по отношению к эталонной операции:

$$B_i = t_i/t_э,$$

где t_i — время реализации i-й операции; $t_э$ — время реализации эталонной операции сложения. Тогда количество условных элементарных операций составит

$$N_э = \sum_{i=1}^{r} B_i Q_i,$$

а общее время их реализации $T_э = N_э t_э$.

Объем информации, используемой алгоритмом, определяется как суммарный объем, необходимый для размещения программной, исходной, промежуточной и выходной информации:

$$И = И_п + И_и + И_в + И_{пр},$$

где И — общий объем информации; $И_п$ — объем программной информации; $И_и$ — объем исходной информации; $И_в$ — объем выходной информации; $И_{пр}$ — объем промежуточной информации.

Тогда обобщенный коэффициент сложности $К_{сл} = N_э/И$, т. е. он характеризуется числом эталонных операций, выполняемых при обработке единицы информации.

В современных ЭВМ распределением памяти занимается операционная система, и пользователь зачастую не знает, какую память он использует.

В современных ЭВМ объем внешней памяти значительно опережает объем оперативной памяти:

$$V_{в.\,п.} \gg V_{о.\,п.},$$

поэтому

$$T = f(N, V_{\text{в. п.}}).$$

Основной путь повышения скорости реализации алгоритма — параллельное выполнение большого числа операций.

Алгоритмы, применяемые на практике, разбивают на два класса: детерминированные и недетерминированные. Детерминированные алгоритмы состоят из операций, результаты из которых определены однозначно, в противном случае — это недетерминированный алгоритм.

Задачей на допустимость называется задача определения хотя бы одного возможного решения.

Задачей на оптимальность считается задача определения допустимого, дающего экстремум функции.

Множество всех задач на допустимость, которые могут быть решены на основе детерминированного алгоритма за полиномиальное время, относятся к подклассу P (полиномиальных алгоритмов).

Подкласс NP — множество всех задач на допустимость, которые могут быть решены с помощью недетерминированного алгоритма, но за полиномиальное время. В NP входят все задачи, для которых известны детерминированные алгоритмы с экспоненциальным временем решения. В технике считается, что $P \subseteq NP$.

Основная задача практики — это исключение или уменьшение перебора при решении конструкторских или технологических задач. Считают, что задача перебора A сводится к задаче перебора B ($A \sim\!\!\rightarrow B$), если метод решения задачи B сводится в метод решения задачи A. Сводимость считается полиномиальной, если такое преобразование происходит за полиномиальное время, в противном случае, сводимость экспоненциальная.

В настоящее время в подклассе NP определено некоторое подмножество $NPC \subseteq NP$. Это подмножество обладает одним замечательным свойством: все проблемы или задачи подкласса NPC могут быть сведены друг к другу. Следовательно, если найдется хотя бы один алгоритм эффективного решения хотя бы одной задачи из NPC, то можно будет эффективно решить все экспоненциальные алгоритмы из этого подкласса.

1.7. Алгоритмические системы решения практических задач. Все операции, выполняемые в алгоритмах, разделяют на две группы:

1 — арифметические операции.

2 — логические операции.

Арифметические операции выполняют непосредственное преобразование информации. Логические операции определяют последовательность выполнения арифметических. На практике распространение получили следующие универсальные схемы алгоритмов.

Приведем один из важных методов ***сортировки вставками***, описанный Корменом Т., Лейзерсоном Ч., Ривестом Р. Он эффективен при сортировке коротких числовых последовательностей. Метод основан на

выборе очередного числа и вставки его в нужное место, сравнивая с имеющимися и передвигаясь справа налево. Исходными данными алгоритма является массив $A[0, \ldots, n]$ (последовательность элементов (чисел) длины n, подлежащая сортировке). Число элементов в массиве A обозначается через length [A]. Последовательность сортируется без дополнительной памяти. Здесь используется входной массив и фиксированное число ячеек памяти.

На рис. П.3 показан алгоритм сортировки Кормена Т., Лейзерсона Ч., Ривеста Р. на примере числовой последовательности $A = (4, 1, 3, 5, 0, 2)$.

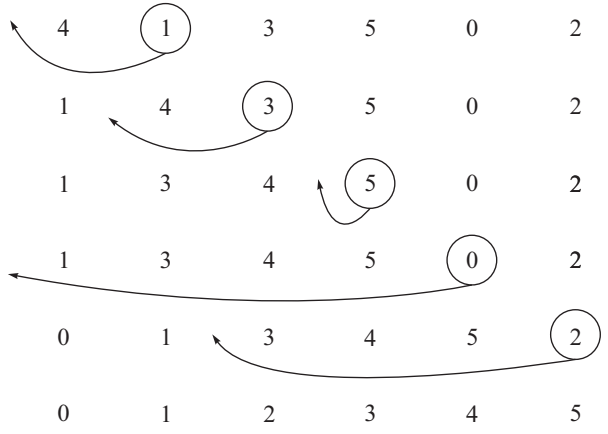


Рис. П.3. Работа процедуры сортировки

Приведем описание алгоритма на верхнем уровне с пояснениями. Такое описание эффективно используется на практике при решении задач информатики и вычислительной техники и называется псевдокодом алгоритма.

ПСЕВДОКОД АЛГОРИТМА СОРТИРОВКИ (Кормена Т., Лейзерсона Ч., Ривеста Р.)

1 `for` $\leftarrow 2$ `to length` $[A]$
2 `do` $key \leftarrow A[j]$
3 «добавить $A[j]$ к отсортированной части $A[0, \ldots, j-1]$»
4 $i \leftarrow j - 1$
5 `while` $i > 0$ `and` $A[i] > key$
6 `do` $A[i+1] \leftarrow A[i]$
7 $i \leftarrow j - 1$
8 $A[i+1] \leftarrow key$

Индекс j указывает выбранный очередной элемент. Часть массива $A[0, \ldots, j-1]$ составляют уже просмотренные элементы, массива $A[j+1, \ldots, n]$ — еще не просмотренные элементы. После выполнения процедуры сортировки массив A упорядочен по возрастанию.

В цикле `for` индекс j проходит массив слева направо. Просматривается элемен $A[j]$ (строка 2 алгоритма) и производится сдвиг стоящих перед ним больших по величине элементов, начиная с $(j-1)$-го вправо, освобождая место для выбранного элемента (строки 4–7). В строке 8 элемент $A[j]$ помещается на свободное место.

Комментарии.

1. Отступ от левого поля указывает на уровень вложения. Например, тело цикла `for` (строка 1) состоит из строк 2–8, а тело цикла `while` (строка 5) содержит строки 6–7.

2. Циклы `while`, `for`, `repeat` и условные конструкции `if-then-else` имеют тот же смысл, что и в языке программирования «Паскаль».

3. Кавычки начинают и заканчивают комментарий (идущий до конца строки).

4. Одновременное присваивание $i \leftarrow j \leftarrow e$ (переменные i и j получают значение e) заменяет два присваивания $j \leftarrow e$ и $i \leftarrow j$ (в этом порядке).

5. Переменные (в данном случае i, j, key) локализуются внутри описанной процедуры (если не оговорено иное).

6. Индекс массива пишется в квадратных скобках, т. е. запись $A[v]$ соответствует v-му элементу в массиве A.

1.8. Структурная схема алгоритма. Еще одной разновидностью графического представления алгоритмов являются *структурные схемы алгоритмов*. При такой записи алгоритмов происходит как бы процесс декомпозиции (т. е. разбиения алгоритма на отдельные этапы-блоки). Блоки изображаются графически в виде геометрических фигур. Каждому такому блоку присваивается собственный номер. Внутри блока в виде формул или словесно записывается информация, характеризующая выполняемые им функции. Блоки соединяются линиями, символизирующими межблочные связи.

Блоки могут описывать элементарные шаги алгоритма или представлять собой части алгоритмов или отдельные алгоритмы.

Правила выполнения структурных схем алгоритмов определены ГОСТом 19.701-90 (ИСО 5807-85) «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения».

Ниже приведен фрагмент текста ГОСТа.

1. Введение

Схема программы отображает последовательность операций в программе и она состоит:

1) из символов процесса, указывающих фактические операции обработки данных (включая символы, определяющие путь, которого следует придерживаться с учетом логических условий);

2) линейных символов, указывающих поток управления;

3) специальных символов, используемых для облегчения написания и чтения схемы.

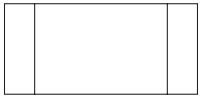
2. Символы

2.1. Процесс.

Символ отображает функцию обработки данных любого вида.

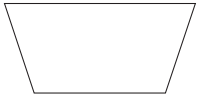
2.2. Предопределенный процесс.

Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, в модуле).



2.3. Ручная операция.

Символ отображает любой процесс, выполняемый человеком.

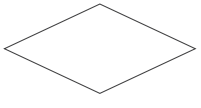


2.4. Подготовка.

Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы).

2.5. Решение.

Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути.



2.6. Линия.

Символ отображает поток данных или управления.

2.7. Соединитель.

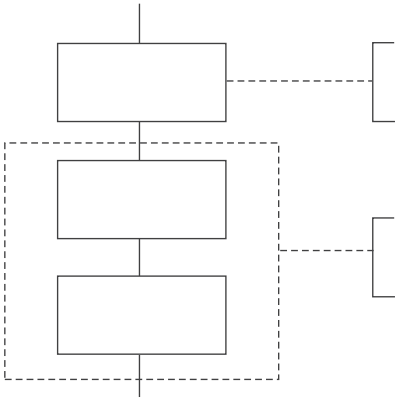
Символ используется для обрыва ЛИНИИ и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.

2.8. Терминатор.

Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец программы, внешнее использование и источник или пункт назначения данных).

2.9. Комментарий.

Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний. Пунктирные линии в символе комментария связаны с соответствующим символом или могут обводить группу символов. Текст комментариев или примечаний должен быть помещен около ограничивающей фигуры.



3. Правила применения символов

3.1. Символ предназначен для графической идентификации функции, которую он обозначает, независимо от текста внутри этого символа.

3.2. Символы в схеме должны быть расположены равномерно. Следует придерживаться разумной длины соединений и минимального числа длинных линий. Символы должны быть, по возможности, одного размера.

3.3. Текст для чтения должен записываться слева направо и сверху вниз независимо от направления потока. Если объем текста, помещаемого внутри символа, превышает его размеры, следует использовать символ комментария.

4. Правила выполнения соединений

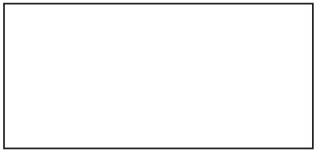
4.1. Направление потока слева направо и сверху вниз считается стандартным.

4.2. В схемах следует избегать пересечения линий. Изменение направления в точках пересечения не допускается.

4.3. Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа.

4.4. При необходимости линии в схемах следует разрывать для избежания излишних пересечений или слишком длинных линий, а также если схема состоит их нескольких страниц. Соединитель в начале разрыва называется внешним соединителем, а соединитель в конце разрыва — внутренним соединителем.

Ссылки к страницам могут быть приведены совместно с символом комментария для их соединителей.



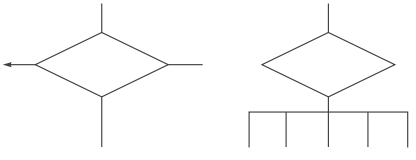
5. Специальные условные обозначения

Несколько выходов из символа следует показывать:

1) несколькими линиями от данного символа к другим символам;

2) одной линией от данного символа, которая затем разветвляется в соответствующее число линий.

Каждый выход из символа должен сопровождаться соответствующими значениями условий, чтобы показать логический путь, который он представляет, с тем, чтобы эти условия и соответствующие ссылки были идентифицированы.



Руководствуясь требованиями данного ГОСТа, легко можно построить структурную схему алгоритма для решения конкретной задачи.

1.9. Словесные структуры алгоритмов. В этом случае по пунктам перечисляются блоки и проверяемые логические условия.

Отметим следующее: словесное представление алгоритма наиболее применимо в диалоговом режиме работы пользователя с ЭВМ.

1.10. Нечеткий алгоритм. *Нечеткий* (*расплывчатый*) *алгоритм* — это упорядоченное множество расплывчатых инструкций, которые при их реализации позволяют получать приближенное решение задач.

Инструкции в расплывчатых алгоритмах делят на три группы.

1-я группа — назначающие инструкции. Например, $x =$ большой или $x =$ вкусный.

2-я группа — расплывчатые высказывания. Например, если x — очень дорогой, то перейти к y, или если x не помещается в комнате, то увеличить размер комнаты.

3-я группа — безусловные активные предложения. Например, немного увеличить x, или сильно увеличить x, или перейти к другому параметру.

Отметим, что все инструкции могут быть расплывчатыми или четкими.

Расплывчатые алгоритмы, которые служат для получения приближенного описания стратегии, называются *расплывчатыми алгоритмами принятия решений*.

К ним, например, можно отнести расплывчатый алгоритм пересечения улицы.

Основными этапами полного построения алгоритма являются:

- постановка задачи,
- построение модели,
- разработка алгоритма,
- реализация алгоритма,
- анализ алгоритма и его сложности,
- составление документации.

Основной метод повышения скорости работы алгоритма — это параллельные вычисления. Согласно известной гипотезе Минского, параллельные машины, выполняющие последовательную программу над n множествами данных, повышают производительность по крайней мере на множитель $O(\log n)$.

Следовательно, можно получить выигрыш на порядок величины, если реализовать алгоритм в параллельной модификации. Это достигается при добавлении n процессоров, увеличении числа выполняемых операций и уменьшении коэффициента использования процессоров.

Приложение 2. Примеры реализации основных генетических операторов

2.1. Пример реализации генетического оператора «элитной селекции».

```
  FunkSelection ( arrPopulation[m], n )
  {
Temp = 0;
```

```
for i = m to 0 do
for j = 0 to m do
 if F( arrPopulation[j] ) < F(arrPopulation[j+1]) then
   Temp = arrPopulation[j];
    arrPopulation[j] = arrPopulation[j+1];
    arrPopulation[j+1] = Temp;
  j = j + 1;
  i = i - 1;
for j = 0 to n do
 arrPopulationN[j] = arrPopulation[j];
  }
```

Функция `FunkSelection` получает в качестве параметров массив `arrPopulation` размера m — начальная популяция, из которой необходимо отобрать лучшие экземпляры; n — число этих экземпляров, $n \leqslant m$.

В первых двух циклах осуществляется упорядочивание представителей популяции по убыванию пропорционально значению заданной целевой функции `F`, при этом используется дополнительная переменная `Temp`.

В последнем цикле выходной массив `arrPopulationN` заполняется первыми n элементами исходного массива с максимальными значениями целевой функции.

Пример. Выбрать из массива значений функции $y = 3x + 2 \rightarrow$ $\rightarrow \max 5$ наилучших.

x	1	2	5	4	8	0	9	3	10	7
y	5	8	17	14	26	2	29	11	32	23

Элементы массива будут отсортированы в порядке убывания: 10, 9, 8, 7, 5, 4, 3, 2, 1, 0; а затем из них будут выбраны 5 первых: 10, 9, 8, 7, 5.

2.2. Пример реализации генетического оператора «турнирной селекции».

```
    FunkSelection ( arrPopulation[m], n )
    {
  do
   TempCol = random ( m );
  if TempCol = m or TempCol = 0 then TempCol = m/2;
  while n > TempCol
  for i = 0 to TempCol do
   arrTempMsv[i] = arrPopulation[random(m)];
  Temp = 0;
  for i = m to 0 do
  for j = 0 to m do
if F(arrTempMsv [j] )<F(arrTempMsv [j+1]) then
   Temp = arrTempMsv [j];
```

```
    arrTempMsv [j] = arrTempMsv [j+1];
    arrTempMsv [j+1] = Temp;
j = j + 1;
i = i - 1;
for j = 0 to n do
    arrPopulationN[j] = arrTempMsv [j];
    }
```

Переменная $1 \leqslant$ `TempCol` $\leqslant m$ задается случайно и представляет собой количество решений, из которых будем выбирать лучшие (размер турнира).

Цикл `do ...while` повторяется до тех пор, пока значение n не будет превышать количество решений, из которых эти n надо выбрать.

Затем формируется массив `arrTempMsv` размерностью `TempCol`, в который заносятся случайно выбранные решения.

Затем производится выбор n решений по элитному принципу из массива `arrTempMsv`.

Пример. Пусть размер турнира 5, и из турнира нам необходимо выбрать 3 решения. Пусть для турнира были выбраны следующие решения: 5, 0, 10, 4, 7. Согласно вышеприведенному алгоритму из них выбираем 3: 10, 7, 5.

2.3. Пример реализации генетического оператора селекции на основе выбора значений целевой функции по принципу «дальнего родства.»

```
  FunkSelection ( arrPopulation[m], n )
  {
Temp = 0;
for i = m to 0 do
for j = 0 to m do
 if F( arrPopulation[j] ) < F(arrPopulation[j+1]) then
   Temp = arrPopulation[j];
    arrPopulation[j] = arrPopulation[j+1];
    arrPopulation[j+1] = Temp;
 j = j + 1;
 i = i - 1;
 for j = 0 to n - 1 do
  arrPopulationN[j] = arrPopulation[j];
  arrPopulationN[j + 1] = arrPopulation[n - j ];
   }
```

Функция `FunkSelection` получает в качестве параметров массив `arrPopulation` размера m — начальная популяция, из которой необходимо отобрать решения, наиболее отличающиеся значениями целевой функцией; n — число этих решений, $n \leqslant m$.

В первых двух циклах осуществляется упорядочивание представителей популяции по убыванию согласно значению заданной целевой функции F, при этом используется дополнительная переменная `Temp`.

В последнем цикле выходной массив `arrPopulationN` заполняется n значениями исходного массива, причем выбор решений ведется с двух сторон, т. е. при $j = 0$ решение выбирается из начала массива, при $j + 1 = 1$ решение выбирается из его конца и т. д.

Пример. Задан массив значений функции $y = 3x + 2$, необходимо выбрать 4 из них.

x	1	2	5	4	8	0	9	3	10	7
y	5	8	17	14	26	2	29	11	32	23

Сначала элементы будут отсортированы в порядке убывания: 10, 9, 8, 7, 5, 4, 3, 2, 1, 0; а затем из них будут выбраны 4 решения: 10, 0; 9, 1.

2.4. Пример реализации процедуры формирования начальной популяции решений по методу «одеяла» и генетического оператора «элитной» селекции и селекции на основе выбора значений целевой функции по принципу «дальнего родства».

Процедура формирования начальной популяции методом «одеяла»:

```
   INPUT Interval;
   INPUT Col;
   for I = 0 to Col - 1
   do
Massiv [I] = Interval/(2*Col) + Interval/(Col)*I;
next I
```

где `Interval` — диапазон изменения значений целевой функции; `Col` — размер популяции; `Massiv` — массив выбранных особей; `I` — переменная цикла.

Пример. Пусть функция рассматривается на интервале от 0 до 100 (`Interval` = 100), размер популяции `Col` = 8. Тогда будут выбраны следующие значения:

I	0	1	2	3	4	5	6	7
Massiv[I]	6	18	30	42	54	66	78	90

Генетический оператор «элитной» селекции:

```
  FunkSelection ( arrPopulation[m], n )
  {
Temp = 0;
for I = m to 0 do
for j = 0 to m do
 if F( arrPopulation[j] ) < F(arrPopulation[j+1]) then
   Temp = arrPopulation[j];
    arrPopulation[j] = arrPopulation[j+1];
```

```
      arrPopulation[j+1] = Temp;
    j = j + 1;
   I = I - 1;
  for j = 0 to n do
      arrPopulationN[j] = arrPopulation[j];
       }
```

Функция `FunkSelection` получает в качестве параметров массив `arrPopulation` размера m — начальная популяция, из которой необходимо отобрать лучшие экземпляры; n — число этих экземпляров, $n \leqslant m$.

В первых двух циклах осуществляется упорядочивание представителей популяции по убыванию согласно значению заданной целевой функции F, при этом используется дополнительная переменная `Temp`.

В последнем цикле выходной массив `arrPopulationN` заполняется первыми n значениями исходного массива, у которых целевая функция максимальна.

Пример. Задан массив значений функции $y = 3x + 2$, необходимо выбрать 5 из них:

x	1	2	5	4	8	0	9	3	10	7
y	5	8	17	14	26	2	29	11	32	23

Элементы массива будут отсортированы в порядке убывания y: 10, 9, 8, 7, 5, 4, 3, 2, 1, 0. Из них будут выбраны 5 первых:

x	10	9	8	7	5
y	32	29	26	23	17

Генетический оператор селекции на основе выбора значений целевой функции по принципу «дальнего родства»:

```
    FunkSelection ( arrPopulation[m], n )
    {
  Temp = 0;
  for i = m to 0 do
  for j = 0 to m do
   if F( arrPopulation[j] ) < F(arrPopulation[j+1]) then
     Temp = arrPopulation[j];
      arrPopulation[j] = arrPopulation[j+1];
      arrPopulation[j+1] = Temp;
   j = j + 1;
   i = i - 1;
   for j = 0 to n - 1 do
      arrPopulationN[j] = arrPopulation[j];
      arrPopulationN[j + 1] = arrPopulation[n - j ];
}
```

Функция `FunkSelection` получает в качестве параметров массив `arrPopulation` размера m — начальная популяция, из которой необходимо отобрать экземпляры, наиболее отличающиеся целевой функцией; n — число этих экземпляров, $n \leqslant m$.

В первых двух циклах осуществляется упорядочивание представителей популяции по убыванию согласно значению заданной целевой функции F, при этом используется дополнительная переменная `Temp`.

В последнем цикле выходной массив `arrPopulationN` заполняется n значениями исходного массива, причем выбор решений ведется с двух сторон, т. е. при $j = 0$ решение выбирается из начала массива, при $j + 1 = 1$ решение выбирается из его конца и т. д.

Пример. Задан массив значений функции $y = 3x + 2$, необходимо выбрать 5 из них.

x	1	2	5	4	8	0	9	3	10	7
y	5	8	17	14	26	2	29	11	32	23

Сначала значения x будут отсортированы в порядке убывания y:

x	10	9	8	7	5	4	3	2	1	0
y	32	29	26	23	17	14	11	8	5	2

Из них будут выбраны 5 следующих:

x	10	0	9	1	8
y	32	2	29	5	26

2.5. Пример реализации процедуры формирования начальной популяции решений по методу «фокусировки» и генетического оператора «турнирной» селекции и селекции на основе выбора значений целевой функции по принципу «близкого родства».

Процедура формирования начальной популяции методом «фокусировки»:

```
INPUT T, n;
A = T*0.25 + random ( T*0.5 );
k = A /(2*n);
for x = 0 to n/2
do
    y = 2*x + k;
    if A + y<T then M[x] = A + y;
    next x
for x = n/2 to n
do
```

```
y = 2*(x-n/2 + 1) + k;
    if A - y >= 0 then M[x] = A - y;
    next x
```

где `T` — размер популяции; `A` — точка фокусировки; `k` — величина смещения; `x` — переменная цикла; `y` — функция смещения; `M` — массив выбранных особей.

После ввода интервала и количества особей, которых необходимо выбрать, выбираем случайным образом точку фокусировки `A`. Устанавливаем величину смещения `k`. В циклах выбираем значения, равноудаленные от точки `A`, и формируем из них массив `M`.

Пример. Пусть размер популяции $T = 100$, количество выбранных особей $n = 8$, $A = 54$. Будут выбраны следующие значения:

I	0	1	2	3	4	5	6	7
M[I]	54	56	58	60	52	50	48	46

Генетический оператор «Турнирной» селекции:

```
  FunkSelection ( arrPopulation[m], n )
  {
do
 TempCol = random ( m );
if TempCol = m or TempCol = 0 then TempCol = m/2;
while n > TempCol
for i = 0 to TempCol do
  arrTempMsv[i] = arrPopulation[random(m)];
  Temp = 0;
 for i = m to 0 do
 for j = 0 to m do
if F(arrTempMsv [j] )<F(arrTempMsv [j+1]) then
     Temp = arrTempMsv [j];
       arrTempMsv [j] = arrTempMsv [j+1];
       arrTempMsv [j+1] = Temp;
  j = j + 1;
  i = i - 1;
 for j = 0 to n do
  arrPopulationN[j] = arrTempMsv [j];
}
```

Переменная `TempCol`, задаваемая случайно — это количество решений, из которых будем выбирать лучшие, должна быть в пределах от 1 до m.

Цикл `do while` повторяется до тех пор, пока значение n не будет превышать количество решений, из которых эти n надо выбрать.

Затем формируется массив `arrTempMsv` размерностью `TempCol`, в который заносятся случайно выбранные решения.

После этого производится выбор n решений элитным отбором из массива `arrTempMsv`.

Пример. Пусть TempCol = 5, необходимо выбрать 3 решения.

x	1	2	5	4	8	0	9	3	10	7
y	5	8	17	14	26	2	29	11	32	23

Были выбраны следующие решения: 5, 0, 10, 4, 7. Согласно вышеприведенному алгоритму, выбираем из них 3: 10, 7, 5.

Генетический оператор селекции на основе выбора значений целевой функции по принципу «близкого родства»:

```
    FunkSelection ( arrMsv[n][m] )
    {
    Temp = random (n);
    for I = 0 to n
    do
Elem = 0;
For j = 0 to m
Do
 if arrMsv [Temp][j] != arrMsv[i][j] and Temp != I then
   Elem = Elem + 1;
    if Elem = 1 then Pos1 = j;
    if Elem = 2 then Pos2 = j;
 if (Elem = 2 and |Pos2 - Pos1| <= 2) or Elem = 1 then
  I = i;
  goto end;
  end
```

где `ArrMsv` — массив решений в двоичном представлении; `n` — количество решений в массиве; `m` — длина каждого решения; `Elem` — количество отличий сравниваемых решений; `Temp` — номер случайно выбранного решения, для которого ищем максимально похожее; `Pos1`, `Pos2` — позиции отличающихся элементов в сравниваемых решениях; `I` — номер найденного решения.

В функцию передается двумерный массив `arrMsv[n][m]`.

Случайным образом выбираем решение с номером `Temp`. В цикле просматриваем все решения и сравниваем их с решением `Temp`. Выбираем решение, для которого `Elem` не превосходит 2, т. е. решения — близкородственные.

Пример.

Номер	Решения в двоичном коде						
№ 1	1	0	0	1	1	1	1
№ 2	1	0	0	0	0	0	1
№ 3	0	0	1	1	1	0	0
№ 4	1	1	0	1	1	0	1
№ 5	1	0	0	0	0	1	0

Пусть задана таблица решений в двоичном коде. Для `Temp` = 2 (1000001) получаем близкородственное решение $I = 5$ (100010)

Temp	1	0	0	0	0	0	1
I	1	0	0	0	0	1	0

***2.6. Пример реализации генетического оператора одноточечного кроссинговера*:**

```
randomize ();
// ввод размера популяции
INPUT n
//заполнение значениями
for intIndexI = 0 to n
for intIndexJ = 0 to 6
arrMassiv[intIndexI][intIndexJ] = random (10);
intIndexJ = intIndexJ + 1
intIndexI = intIndexI + 1
// выбор 2 решений для скрещивания
do
NumberOne = random ( n );
NumberTwo = random ( n );
while ( NumberOne = NumberTwo );
// выбор точки разрыва
Point = random ( 6 - 2 );
// кроссинговер
for intIndex = 0 to 6
// до точки разрыва
if ( intIndex < Point )
{
arrMassivResult[0][intIndex]
       = arrMassiv[NumberOne][intIndex];
arrMassivResult[1][intIndex]
       = arrMassiv[NumberTwo][intIndex];
}
// после точки разрыва
if ( intIndex >= Point )
{
arrMassivResult[0][intIndex]
       = arrMassiv[NumberTwo][intIndex];
arrMassivResult[1][intIndex]
       = arrMassiv[NumberOne][intIndex];
}
intIndex = intIndex + 1;
}
```

где `ArrMassiv[n][6]` — начальная популяция размерностью в n особей; `NumberOne`, `NumberTwo` — номера скрещиваемых особей; `Point` — точка разрыва; `ArrMassivResult[2][6]` — два потомка.

Пример. Пусть размер популяции $n = 4$, `NumberOne` $= 1$, `NumberTwo` $= 4$, `Point` $= 2$.

`ArrMassiv`:

i \ j	1	2	3	4	5	6
1	0	5	8	4	3	1
2	0	0	8	6	1	5
3	1	1	7	3	5	5
4	0	5	9	4	4	6

arrResult[1]	0	5	9	4	4	6
arrResult[2]	0	5	8	4	3	1

***2.7. Пример реализации генетического оператора универсального кроссинговера*:**

```
randomize ();
// размер популяции
INPUT n;
// Заполнение массива значениями
for intIndexI = 0 to n
{
for intIndexJ = 0 to 6
{
arrMassiv[intIndexI][intIndexJ] = random (2);
intIndexJ = intIndexJ + 1
}
intIndexI = intIndexI + 1
}
// Выбор 2 решений для скрещивания
do
{
NumberOne = random ( n );
NumberTwo = random ( n );
}
while ( NumberOne = = NumberTwo );
// маска
for i = 0 to 6
{
arrMaska[i] = random ( 2 );
i = i + 1;
}
// Кроссинговер
for intIndex = 0 to 6
```

```
{
// если - 1
if ( arrMaska[intIndex] )
{
arrMassivResult[0][intIndex]
       = arrMassiv[NumberOne][intIndex];
arrMassivResult[1][intIndex]
       = arrMassiv[NumberTwo][intIndex];
}
// если - 0
else
{
arrMassivResult[0][intIndex]
       = !arrMassiv[NumberOne][intIndex];
arrMassivResult[1][intIndex]
       = !arrMassiv[NumberTwo][intIndex];
}
intIndex = intIndex + 1
}
```

где `ArrMassiv[n][6]` — начальная популяция размерностью в n особей; `NumberOne, NumberTwo` — номера скрещиваемых особей; `arrMaska[6]` — маска для получения потомков; `arrMassivResult[2][6]` — два потомка.

Пример. Пусть задана популяция `arrMassiv[4][6]`:

i \ j	1	2	3	4	5	6
1	0	1	0	1	1	1
2	1	0	0	1	0	1
3	1	1	1	0	0	1
4	0	1	0	0	1	1

```
NumberOne = 2
NumberTwo = 3
arrMaska[6] = {1, 1, 1, 1, 0, 1}
```

arrResult[1]	1	0	0	1	1	1
arrResult[2]	1	1	1	0	1	1

***2.8. Пример реализации генетического оператора упорядочивающего кроссинговера*:**

```
randomize ();
INPUT n;
```

```
for intIndexI = 0 to n
{
for intIndexJ = 0 to 6
{
arrMassiv[intIndexI][intIndexJ] = random (10);
intIndexJ = intIndexJ + 1
}
intIndexI = intIndexI + 1
}
do
{
NumberOne = random ( n );
NumberTwo = random ( n );
}
while ( NumberOne == NumberTwo );
// позиция разрыва
Point = random ( 6 - 2 );

for intIndex = 0 to Point
{
// до точки разрыва
arrMassivResult[0][intIndex]
       = arrMassiv[NumberOne][intIndex];
arrMassivResult[1][intIndex]
       = arrMassiv[NumberTwo][intIndex];
intIndex = intIndex + 1;
}
// упорядочивание по возрастанию после точки разрыва
//в обоих решениях
for i = 5; to Point
for j = Point to 6
{
if (arrMassiv[NumberOne][j] > arrMassiv[NumberOne][j + 1])
{
Temp = arrMassiv[NumberOne][j];
arrMassiv[NumberOne][j] = arrMassiv NumberOne][j + 1];
arrMassiv[NumberOne][j + 1] = Temp;

}
if (arrMassiv[NumberTwo][j] > arrMassiv[NumberTwo][j + 1])
{
Temp = arrMassiv[NumberTwo][j];
arrMassiv[NumberTwo][j] = arrMassiv[NumberTwo][j + 1];
arrMassiv[NumberTwo][j + 1] = Temp;

}
j = j + 1;
i = i - 1;
}
// копирование уже отсортированных после точки разрыва
```

```
//генов
for intIndex = Point to 6
{
arrMassivResult[0][intIndex]
       = arrMassiv[NumberOne][intIndex];
arrMassivResult[1][intIndex]
       = arrMassiv[NumberTwo][intIndex];
intIndex = intIndex + 1;

}
```

Пример. Для популяции, заданной в первом примере:

arrResult[1]	0	5	1	3	4	8
arrResult[2]	0	5	4	4	9	9

***2.9. Пример реализации генетического оператора двухточечного кроссинговера*:**

```
randomize ();
// ввод размера популяции
INPUT n
// заполнение массива значениями
for intIndexI = 0 to n
{
for intIndexJ = 0 to 6
{
arrMassiv[intIndexI][intIndexJ] = random (10);
intIndexJ = intIndexJ + 1;
}
intIndexI = intIndexI + 1;
}
// выбор двух решений
do
{
NumberOne = random ( 6 );
NumberTwo = random ( 6 );
}
while ( NumberOne == NumberTwo );
// выбор точек разрыва
do
{
Point1 = random ( 6 - 2 );
Point2 = random ( 6 - 2 );
}
while ( Point1 >= Point2 );
// кроссинговер
for intIndex = 0 to 6
{
```

```
// до первой точки разрыва
if ( intIndex < Point1 )
{
arrMassivResult[0][intIndex]
       = arrMassiv[NumberOne][intIndex];
arrMassivResult[1][intIndex]
       = arrMassiv[NumberTwo][intIndex];
}
// между точками разрыва
if ( (intIndex >= Point1) {\&}{\&} (intIndex < Point2) )
{
arrMassivResult[0][intIndex]
       = arrMassiv[NumberTwo][intIndex];
arrMassivResult[1][intIndex]
       = arrMassiv[NumberOne][intIndex];
}
// после второй точки разрыва
if ( intIndex >= Point2 )
{
arrMassivResult[0][intIndex]
       = arrMassiv[NumberOne][intIndex];
arrMassivResult[1][intIndex]
       = arrMassiv[NumberTwo][intIndex];
}
intIndex = intIndex + 1
}
```

где `ArrMassiv[n][6]` — начальная популяция размерностью в n особей; `NumberOne, NumberTwo` — номера скрещиваемых особей; `Point1, Point2` — точки разрыва; `ArrMassivResult[2][6]` — два потомка.

Пример. Пусть размер популяции $n = 4$, `NumberOne` $= 1$, `NumberTwo` $= 4$, `Point1` $= 2$, `Point2` $= 4$.

`ArrMassiv`:

i \ j	1	2	3	4	5	6
1	0	5	8	4	3	1
2	0	0	8	6	1	5
3	1	1	7	3	5	5
4	0	5	9	4	4	6

`aarResult[1]`	0	5	9	4	3	1
`aarResult[2]`	0	5	8	4	4	6

***2.10. Пример реализации генетического оператора циклического кроссинговера*:**

```
// ввод двух решений
for i = 0 to 2
{
for j = 0 to 6
{
INPUT arrMassiv[i][j];
j = j + 1;
}
i = i + 1;
}
for i = 0 to 6
{
arrResult[0][arrMassiv [0][i]] = arrMassiv [1][i];
arrResult[1][arrMassiv [1][i]] = arrMassiv [0][i];
i = i + 1;
}
```

где `arrMassiv[2][6]` — два введенных решения; `arrResult [2][6]` — два потомка.

Пример.

arrMassiv[1]	0	5	1	3	4	2
arrMassiv[2]	0	5	3	2	1	4

arrResult[1]	0	3	4	2	1	5
arrResult[2]	0	4	3	1	2	5

***2.11. Пример реализации генетического оператора «жадного» кроссинговера*:**

```
m_iRandP = random(m_iNumRe);
// берем от точки разрыва левый элемент
nInLeft = Index[0][m_iRandP];
k=-1;
j=0;
sum = 1000;
bNetPovtorov = true;
// устанавливаем первый элемент
Otvet[0] = nInLeft;
for u = 0 to m_iRandX-1
{
sum = 1000;
k = 0;
for i = 0; to m_iRandX
{
```

```
for g=0 to m_iNumRe
{
bNetPovtorov = true;
if (nInLeft = Index[i][g])
{
// находим стоимости путей
if ( g + 1 != m_iNumRe)
{
if (sum > m_piGrid[nInLeft][Index[i][g+1]])
{
for j=0 to u+1
{
if (Index[i][g+1] == Otvet[j])
bNetPovtorov = false;
i = i + 1
}
// если нет повторов, добавляем в решение
if (bNetPovtorov)
{
sum = m_piGrid[nInLeft][Index[i][g+1]];
Otvet[u+1] = Index[i][g+1];
k = Index[i][g+1];
break;
}
}
}
else
{
if (sum > m_piGrid[nInLeft][Index[i][0]])
{
for j=0 to u
{
if (Index[i][0] == Otvet[j])
bNetPovtorov = false;
j = j + 1;
}
if (bNetPovtorov)
{
sum = m_piGrid[nInLeft][Index[i][0]];
Otvet[u+1] = Index[i][0];
k = Index[i][0];
break;
}
}
}
}
g = g + 1;
}
i = i + 1;
}
```

```
nInLeft = k;
u = u + 1;
}
```

где `Index` — массив начальной популяции; `nInLeft` — значение левого элемента от точки разрыва; `m_iRandP` — точка разрыва; `sum` — наименьшее значение стоимости пути в данной хромосоме; `bNetPovtorov` — проверяет наличие повторов элементов в хромосоме; `Otvet` — искомая хромосома; `m_piGrid` — матрица смежности с весами.

Пример. Пусть матрица смежности для 4 элементов имеет вид:

	a	b	c	d
a	—	2	4	5
b	2	—	8	1
c	4	8	—	3
d	5	1	3	—

Начальная популяция $P = 3$, $F(P[I])$ — стоимость пути в данном решении:

№	$F(P[I])$	1	2	3	4
$P[1]$	14	a	c	b	d
$P[2]$	11	c	a	d	b
$P[3]$	17	b	c	a	d

Пусть была выбрана точка между c и b в первом решении, тогда в результате работы алгоритма будет получено искомое решение с $F(P^*) = 11$:

P^*	c	a	d	b

2.12. Пример реализации простого генетического алгоритма.

Приведем сначала словесное описание алгоритма.

1. Сконструировать начальную популяцию альтернативных решений. Ввести точку отсчета поколений $t = 0$. Вычислить приспособленность каждой хромосомы (альтернативного решения) в популяции, а затем среднюю приспособленность всей популяции.

2. Установить $t = t + 1$. Произвести выбор 2 родителей для реализации оператора кроссинговера. Он будет выполняться случайным образом пропорционально приспособленности родителей.

3. Сформировать генотип потомков. Для этого с заданной вероятностью выполнить оператор кроссинговера над генотипом выбранных хромосом. Далее с вероятностью 0,5 выбрать один из потомков Pi и сохранить как член новой популяции. После этого к Pi последовательно применяется оператор инверсии, а затем мутации с заданными вероятностями.

4. Определить количество хромосом для исключения из популяции. Текущую популяцию обновить заменой выбранных хромосом на потомков.

5. Провести определение приспособленности и пересчет средней приспособленности всей популяции.

6. Если $t = t_{\text{заданному}}$, завершить работу, иначе перейти к пункту 2.

Псевдокод алгоритма

1. Основной цикл

```
INPUT T0, n, m       // ввод числа генераций и размера
                                                популяции
T = 0;                // установить текущую генерацию
NewDecisions[2][n];   // массив новых решений
NewDecisionOne[n];    // одно новое решение
ArrPopulation[m][n]; // массив, содержащий популяцию
FirstPopulation ();   // формирование начальной популяции
while T!=T0          // основной цикл алгоритма
{
OKFunc ();            // кроссинговер
MutationFunc ();      // оператор мутации для нового решения
Recombination ();     // оператор рекомбинации
}
```

2. Формирование начальной популяции

```
FirstPopulation ()
{
    INPUT ValueFirst, ValueLast; // ввод диапазона значений,
                                 // из которых будут
                                 // выбираться решения,
                                 // ValueLast >ValueFirst
    SumZF = 0;                   // суммарное значение Цф
// выбор элементов из заданного интервала
      for I = 0 to n do
      for J = 0 to m do
      ArrPopulation[I][J] - random (ValueLast - ValueFirst);
      J = J+1;
   I = I+1;
}
```

3. Оператор кроссинговера (ОК)

```
OKFunc ()
{
// выбор 2 разных решений для скрещивания
do
     NumberOne = random (n);
```

```
     NumberTwo = random (n);
while (NumberOne = NumberTwo);
// выбор точки разрыва
Point = random (m - 1);
// кроссинговер
for I = 0 to m
// до точки разрыва
     if(I < Point)
     {
        NewDecisions [0][I]
             = ArrPopulation [NumberOne][I];
        NewDecisions [1][I]
             = ArrPopulation [NumberTwo][I];
     }
     // после точки разрыва
     if (Index >= Point)
     {
        NewDecisions [0][I]
             = ArrPopulation [NumberTwo][I];
        NewDecisions [1][I]
             = ArrPopulation [NumberOne][I];
     }
     I = I + 1;
// выбор из двух потомков одного
NewDecisionOne = NewDecisions[random (2)];
}
```

4. Оператор мутации (ОМ)

```
MutationFunc ()
{
    // выбор гена в хромосоме для перестановки
    GeneIndex = random (m);
    // если выбранный ген - самый правый
    if GeneIndex = m - 1 then
    {
       // переставляем влево
       Temp = NewDecisionOne [GeneIndex - 1];
       NewDecisionOne[GeneIndex - 1]
                 = NewDecisionOne[GeneIndex];
NewDecisionOne[GeneIndex] = Temp;
    }
    // иначе переставляем вправо
    else
    {
       Temp = NewDecisionOne[GeneIndex];
       NewDecisionOne[GeneIndex]
                 = NewDecisionOne[GeneIndex + 1];
       NewDecisionOne[GeneIndex + 1] = Temp;
    }
}
```

5. Оператор рекомбинации(ОРек)

```
Recombination ()
{
    MinZF = 10000;
IndexMinZF;
// поиск решения с минимальной целевой функции
// по заданному правилу
for I = 0 to n do
        if ValueZF(ArrPopulation[I]) < MinZF then
           MinZF = ValueZF(ArrPopulation[I]);
           IndexMinZF = I;
        // замена найденного решения новым
        ArrPopulation[IndexMinZF] = NewDecisionOne;
}
```

Приложение 3. Задания к лабораторным работам

Лабораторная работа № 1
Основные положения теории генетического поиска (2 часа)

1. Цель работы

Ознакомиться с основными терминами и положениями теории генетического поиска.

2. Порядок выполнения лабораторной работы

2.1. Изучить теоретическое введение.

2.2. Последовательно выполнить все задания к лабораторной работе.

2.3. Проверить правильность выполнения заданий не менее, чем на пяти примерах.

2.4. Оформить отчет по лабораторной работе.

2.5. Ответить на контрольные вопросы.

3. Задания к лабораторной работе

3.1. Написать программу, реализующую различные стандартные виды стратегий создания начальной популяции. Продемонстрировать и объяснить работу программы на примерах.

3.2. Написать программу, реализующую различные варианты развития популяции: микро- и макроэволюцию. Продемонстрировать и объяснить работу программы на примерах.

3.3. Написать программу, позволяющую синтезировать различные виды хромосом (бинарные, числовые, векторные и т. д.). Продемонстрировать и объяснить работу программы на примерах.

3.4. На основе полученных знаний предложить новую модификацию стратегии создания начальной популяции. Написать программу, реализующую разработанную стратегию. Продемонстрировать и объяснить работу программы на примерах.

3.5. Провести статистические испытания написанных программ. Построить тестовые примеры. Провести сравнение и построить диаграммы и графики зависимости времени решения от значений целевой функции.

4. Содержание отчета по лабораторной работе

4.1. Название и цель работы.

4.2. Задания.

4.3. Примеры решений.

Контрольные вопросы

1. Назовите методы решения оптимизационных задач, их достоинства и недостатки.
2. В чем состоит отличие генетических алгоритмов от других методов решения оптимизационных задач?
3. Определите основные термины теории генетического поиска:
   а) популяция;
   б) размер популяции;
   в) число поколений;
   г) лидер и аутсайдер популяции;
   д) хромосома;
   е) ген;
   ж) локус;
   з) аллель;
   и) фенотип.
4. Перечислите механизмы передачи наследственной информации и охарактеризуйте кратко каждый из них.
5. Каковы основные свойства панмиктической популяции (полученной путем свободного скрещивания)?
6. Перечислите основные стратегии создания начальной популяции и укажите их основные достоинства и недостатки.
7. Назовите основные виды хромосом, применяемых при решении технических задач.
8. Какие способы преобразования наследственной информации вам известны? Назовите основные особенности каждого из этих способов.
9. Опишите воздействие мутации на наследственность.
10. Опишите основные варианты развития популяции: микро- и макроэволюцию. Их основные особенности и сущность.

Лабораторная работа № 2
Изучение основных операторов кроссинговера (4 часа)

1. Цель работы

Ознакомиться со схемами выполнения операции кроссинговера.

2. Порядок выполнения лабораторной работы

2.1. Изучить теоретическое введение.

2.2. Последовательно выполнить все задания к лабораторной работе.

2.3. Проверить правильность выполнения задания на пяти примерах.

2.4. Оформить отчет по лабораторной работе.

2.5. Ответить на контрольные вопросы.

3. Задания к лабораторной работе

3.1. Написать программу, реализующую работу основных операторов кроссинговера:

а) одноточечного оператора кроссинговера;

б) двухточечного оператора кроссинговера;

в) трехточечного оператора кроссинговера;

г) универсального оператора кроссинговера;

д) упорядочивающего одно- и двухточечный операторы кроссинговера;

е) частично-соответствующего одно- и двухточечному операторам кроссинговера;

ж) циклического оператора кроссинговера;

з) «жадного» оператора кроссинговера;

и) одно- двух- и трехточечного операторов кроссинговера на основе принципа «золотого сечения» и чисел Фибоначчи.

Продемонстрировать и объяснить работу программы на примерах.

3.2. На основе полученных знаний разработать новые модифицированные схемы выполнения специальных операторов кроссинговера. Разработанные схемы реализовать в виде программы. Продемонстрировать и объяснить работу программы на примерах.

3.3. Построить графики зависимости целевой функции от типа применяемого оператора кроссинговера числа генераций и от времени получения локально-оптимальных знчений целевой функции.

4. Содержание отчета по лабораторной работе

4.1. Название и цель работы.

4.2. Задания.

4.3. Примеры решений.

Контрольные вопросы

1. Приведите примеры выполнения вышеперечисленных операторов кроссинговера.
2. Дайте определения следующих понятий:
   а) схемы;
   б) длины схемы;
   в) определяющей длины;
   г) порядка схемы.
3. Что такое вероятность применения кроссинговера?
4. Запишите и объясните формулу расчета вероятности выживания схемы при кроссинговере.

5. Запишите формулу вероятности выживания схемы при двухточечном операторе кроссинговера.
6. Сравните вероятность выживания схемы при использовании стандартных и специальных операторов кроссинговера. Как можно оценить вероятность выживания схемы после выполнения специальных операторов кроссинговера?

Лабораторная работа № 3
Изучение основных операторов мутации (4 часа)

1. Цель работы

Ознакомиться с основными операторами мутации и их ролью в процессе генетического поиска.

2. Порядок выполнения лабораторной работы

2.1. Изучить теоретическое введение.

2.2. Последовательно выполнить все задания к лабораторной работе.

2.3. Проверить правильность выполнения не менее чем на 10 примерах.

2.4. Оформить отчет по лабораторной работе.

2.5. Ответить на контрольные вопросы.

3. Задания к лабораторной работе

3.1. Написать программу, реализующую работу основных операторов мутации и их разновидностей для различных видов хромосом и схем:

а) простая мутация;

б) точечная мутация;

в) мутация обмена (одно- и двухточечная);

г) мутация на основе принципа «золотого сечения»;

д) мутация на основе чисел Фибоначчи;

е) инверсия;

ж) транслокация;

з) делеция.

Продемонстрировать и объяснить работу программы на примерах.

3.2. Построить графики зависимостей целевой функции от типа применяемого оператора мутации числа генераций и от времени работы оператора.

3.3. Провести статистические испытания программы. Сравнить между собой различные виды операторов.

4. Содержание отчета по лабораторной работе

4.1. Название и цель работы.

4.2. Задания.

4.3. Примеры решений.

Контрольные вопросы

1. Объясните роль оператора мутации в генетическом алгоритме.
2. Опишите основные операторы и разновидности мутаций.
3. Дайте определение мутации, аллеля, генотипа.
4. Каким образом сказывается влияние оператора мутации на возможность выживания схемы?
5. Запишите и объясните формулу расчета вероятности выживания схемы после выполнения операции мутации.
6. Назовите типы мутантов.
7. Какие типы мутантов оказываются полезными при решении технических задач и почему?
8. Приведите описание оператора инверсии по Холланду.
9. Покажите, каким образом определяется вероятность выживания схемы после применения оператора инверсии.

Лабораторная работа № 4
Изучение операторов селекции и отбора (2 часа)

1. Цель работы

Ознакомиться с основными принципами и операторами селекции и отбора в генетическом алгоритме.

2. Порядок выполнения лабораторной работы

2.1. Изучить теоретическое введение.

2.2. Последовательно выполнить все задания к лабораторной работе.

2.3. Проверить правильность выполнения заданий не менее, чем на пяти примерах.

2.4. Оформить отчет по лабораторной работе.

2.5. Ответить на контрольные вопросы.

3. Задания к лабораторной работе

3.1. Написать программу, реализующую различные виды селекции. Продемонстрировать и объяснить работу программы на примерах.

3.2. Написать программу, реализующую различные виды отбора. Продемонстрировать и объяснить работу программы на примерах.

3.3. На основе полученных знаний предложить новые модификации операторов селекции и отбора. Написать программу, реализующую разработанные схемы. Продемонстрировать работу программы на примерах.

3.4. Провести статистические испытания программы. Сравнить между собой различные виды селекции. Построить графики и диаграммы зависимостей целевой функции от различных параметров селекции.

4. Содержание отчета по лабораторной работе

4.1. Название и цель работы.
4.2. Задания.
4.3. Примеры решений.

Контрольные вопросы

1. Дайте определение селекции.
2. Перечислите основные виды селекции. Приведите примеры реализации основных видов селекции.
3. Поясните сущность отбора и его реализации в генетическом алгоритме.
4. Перечислите основные виды отбора, их сущность, особенности и достоинства.
5. Опишите различные схемы отбора, реализуемые при решении оптимизационных задач, и приведите примеры.
6. Опишите эволюционный процесс как процесс развития популяции.
7. Объясните значение терминов: лидер популяции, аутсайдер популяции.

Лабораторная работа № 5
Построение простого генетического алгоритма (4 часа)

1. Цель работы

Ознакомиться с различными схемами выполнения простого генетического алгоритма. Изучить применение фундаментальной теоремы генетических алгоритмов.

2. Порядок выполнения лабораторной работы

2.1. Изучить теоретическое введение.

2.2. Последовательно выполнить все задания к лабораторной работе.

2.3. Проверить правильность выполнения заданий не менее, чем на пяти примерах.

2.4. Оформить отчет по лабораторной работе.

2.5. Ответить на контрольные вопросы.

3. Задания к лабораторной работе

3.1. Выполнить одно из заданий к лабораторной работе и построить на основе описаний генетического алгоритма Холланда, Голдберга и Девиса простой генетический алгоритм вычисления экстремумов функции:

а) $\min(5x^3 - 4)$ на интервале $[1, 2, 3, 4, 5]$;
б) $\max(2x^4 + 12)$ на интервале $[0, 1, 2, 3, 4]$;
в) $\max(3x^3 - 2x + 5)$ на интервале $[1{-}10]$;
г) $\min(5x^2 + 2x - 10)$ на интервале $[5{-}15]$;
д) $\min(x^3 + 2x^2)$ на интервале $[5, 6, 7, 8, 9]$;

е) $\min(x^2 - 5x + 6)$ на интервале $[3, 4, 5, 6, 7]$;
ж) $\max(x^2 + 20x - 34)$ на интервале $[8{-}12]$;
з) $\max(x^2 + 0{,}1x - 23)$ на интервале $[9{-}14]$.

Размер начальной популяции — 10, формирование начальной популяции — по выбору пользователя, вероятность кроссинговера — 70 %, вероятность мутации — 20 %, число генераций — не менее 10.

3.2. Написать программу, реализующую различные схемы простого генетического алгоритма с возможностью задания пользователем размера популяции, числа генераций и вероятностей оператора кроссинговера, оператора мутации и оператора инверсии.

3.3. Сравнить результаты работы простого генетического алгоритма по Голдбергу, Холланду и Девису на основе статистических испытаний.

3.4. Построить графики зависимостей целевой функции от числа генераций алгоритма, времени решения, размера популяции, вероятностей применения операторов и т. д.

4. Содержание отчета по лабораторной работе

4.1. Название и цель работы.
4.2. Задание.
4.3. Структурная схема алгоритмов.
4.4. Примеры решений.

Контрольные вопросы

1. Поясните сущность термина «генетические алгоритмы».
2. Выделите основные отличительные особенности генетического алгоритма.
3. Что такое простой генетический алгоритм?
4. Опишите оператор репродукции в простом генетическом алгоритме.
5. Приведите пример работы оператора репродукции.
6. Запишите формулу расчета вероятности выживания схемы после выполнения оператора репродукции.
7. Запишите фундаментальную теорему генетических алгоритмов и пример вычисления на основе фундаментальной теоремы генетического алгоритма.
8. Покажите на примере вычисление $m(H, t + 1)$ для всего простого генетического алгоритма.
9. Покажите на примере вычисление $P(d)$.
10. Покажите на примере вычисление $P(s)$.
11. Нарисуйте структурную схему простого генетического алгоритма по Холланду.
12. Нарисуйте структурную схему простого генетического алгоритма по Голдбергу.
13. Нарисуйте структурную схему простого генетического алгоритма по Девису.
14. Объясните смысл понятия «строительный блок».

Лабораторная работа № 6
Статистическое исследование результатов работы генетического алгоритма (2 часа)

1. Цель работы

Ознакомиться с основными методами проведения статистических исследований экспериментальных данных.

2. Порядок выполнения лабораторной работы

2.1. Изучить теоретическое введение.

2.2. Последовательно выполнить все задания к лабораторной работе.

2.3. Проверить правильность выполнения заданий не менее чем на пяти примерах.

2.4. Оформить отчет по лабораторной работе.

2.5. Ответить на контрольные вопросы.

3. Задания к лабораторной работе

3.1. Для программ, реализованных в предыдущих лабораторных работах, провести статистическую обработку полученных данных.

3.2. Провести последовательно серии по 50 экспериментов в каждой, в соответствии с заданным шагом, изменяя один из генетических параметров (норма кроссинговера, мутации; размер популяции, число поколений) и фиксируя все остальные.

3.3. Построить тестовые примеры (бенчмарки) и провести количественные и качественные сравнения полученных результатов.

3.4. Построить на основе полученных экспериментальных данных гистограммы и диаграммы накопленных частот.

3.5. Сделать выводы о законе распределения случайной величины (функции стоимости).

4. Содержание отчета по лабораторной работе

4.1. Название и цель работы.

4.2. Задания.

4.3. Описание серий экспериментов.

4.4. Таблицы с результатами проведенных исследований.

4.5. Гистограммы и диаграммы накопленных частот.

4.6. Выводы.

Контрольные вопросы

1. Назовите основные задачи математической статистики.
2. Перечислите основные статистические характеристики.
3. Дайте определение случайной величины. Чем отличается дискретная случайная величина от непрерывной?
4. Что такое частота случайной величины и как она связана с вероятностью?
5. Что такое ряд распределения случайной величины и как он задается?
6. Как задается условие нормировки?

7. Чем отличается ряд распределения от интегральной функции распределения?
8. Что такое плотность распределения и как она связана с интегральной функцией распределения?
9. Определите понятия генеральной совокупности, выборки и ее объема.
10. Что такое математическое ожидание, дисперсия?
11. Как определяются оценки математического ожидания и дисперсии?
12. Каким образом составляется гистограмма, диаграмма накопленных частот для непрерывных и дискретных случайных величин?
13. В чем состоит отличие гистограммы от диаграммы накопленных частот?
14. Что такое доверительная вероятность, доверительный интервал?
15. Что такое интервальная оценка и чем она отличается от точечной?
16. Что такое нормальный закон распределения? Поясните методику его расчета.
17. Каким образом производится проверка правдоподобия гипотез соответствия эмпирического закона распределения случайной величины теоретическому? Поясните методику проверки.
18. Поясните, каким образом изменяются результаты генетического алгоритма при изменении нормы кроссинговера и мутации.
19. Опишите влияние вероятностей оператора кроссинговера, оператора мутации и ОИ на качество и скорость получения решений генетического алгоритма.

Примеры выполнения лабораторных работ

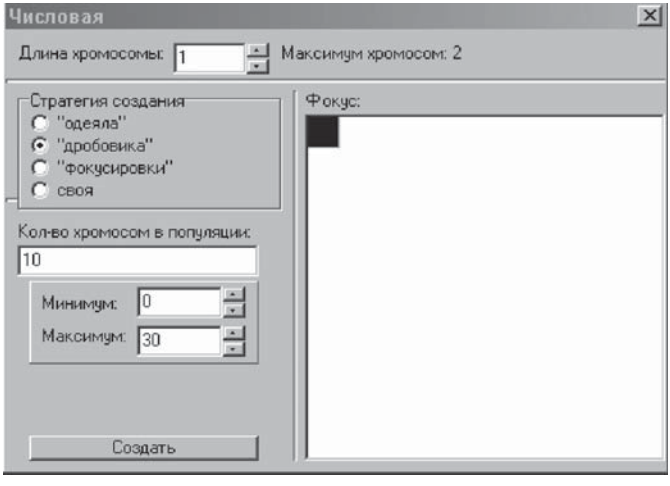


Рис. П.4. Пример окна задания типа хромосом и стратегии создания популяции

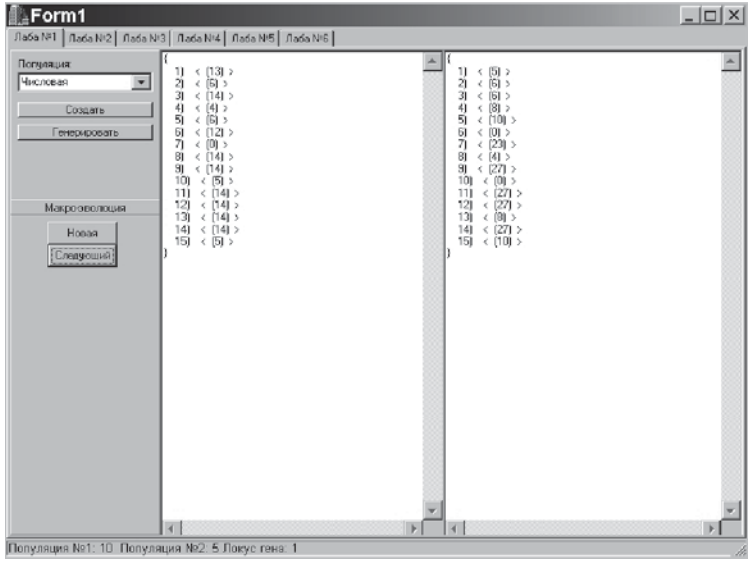


Рис. П.5. Пример окна задания стартовой популяции

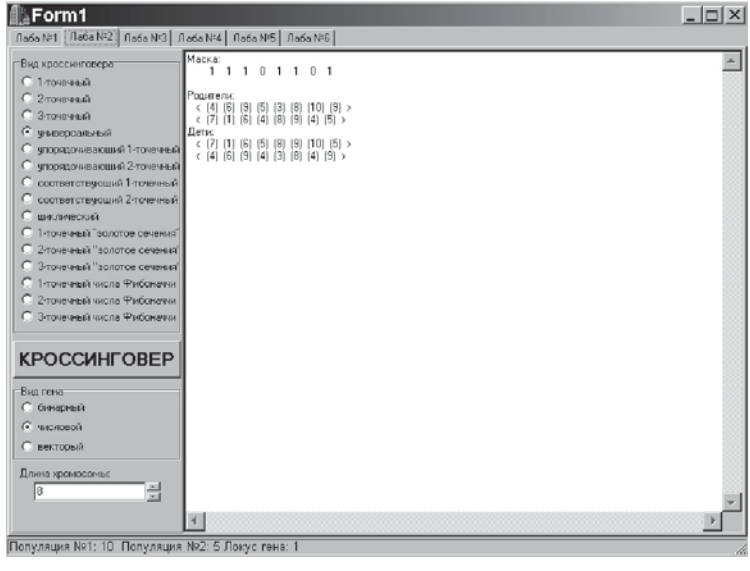


Рис. П.6. Пример окна, отображающего реализацию различных операторов кроссинговера

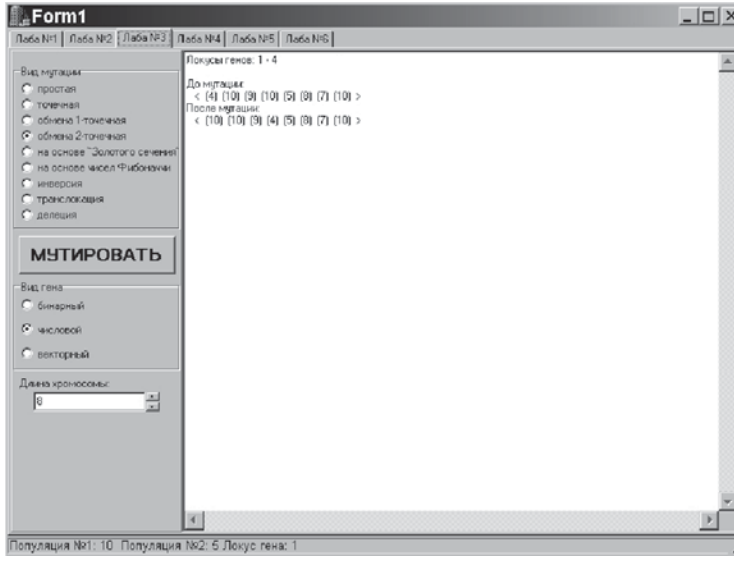


Рис. П.7. Пример окна, отображающего реализацию различных операторов мутации

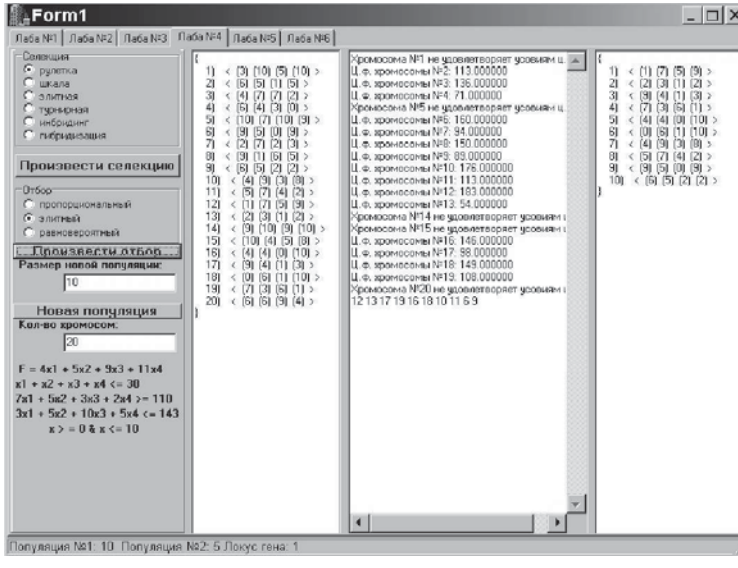


Рис. П.8. Пример окна, отображающего реализацию различных типов селекции

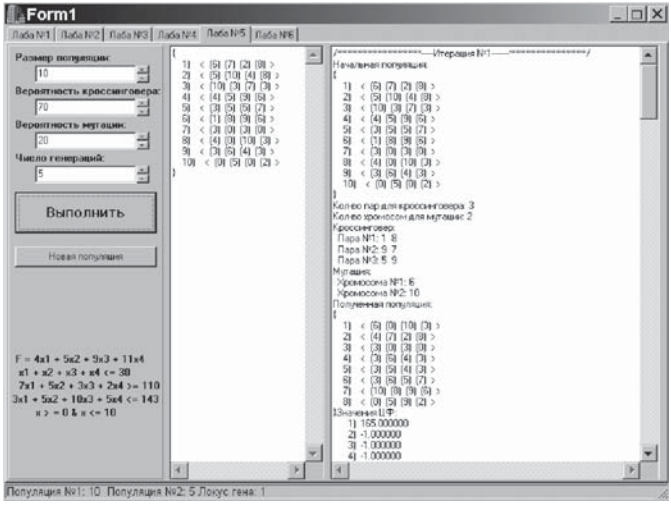


Рис. П.9. Пример работы простого генетического алгоритма по поиску максимума функции

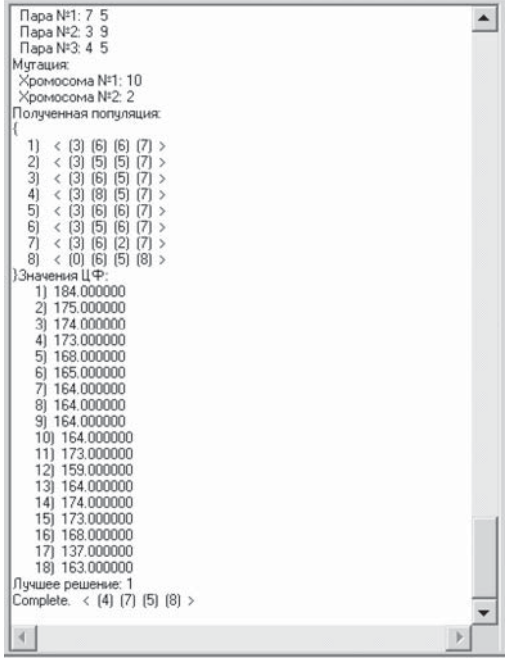


Рис. П.10. Пример окна вывода результатов работы простого генетического алгоритма

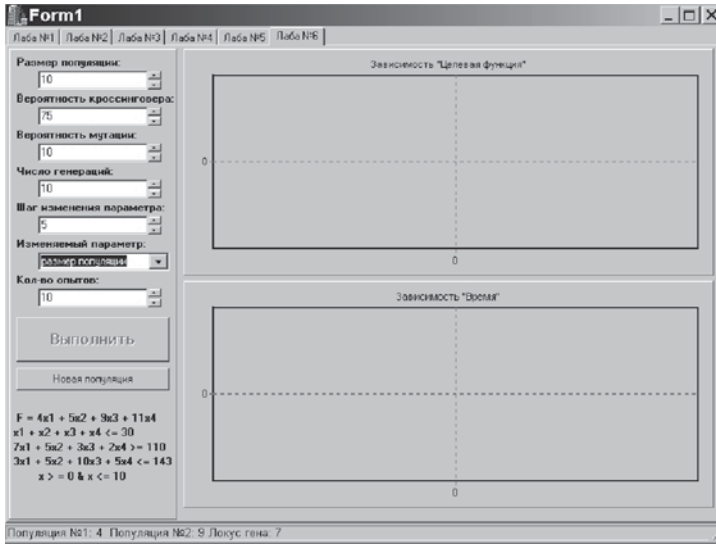


Рис. П.11. Пример окна задания параметров
простого генетического алгоритма

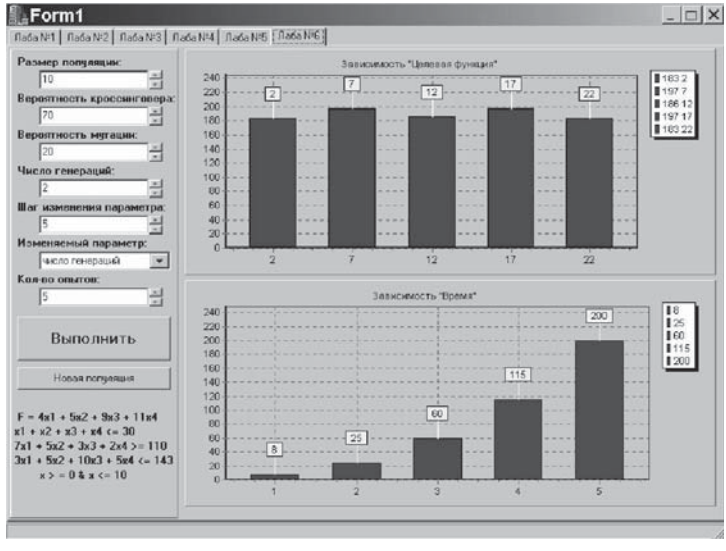


Рис. П.12. Пример окна вывода зависимостей результатов работы
простого генетического алгоритма
от заданных параметров (начало)

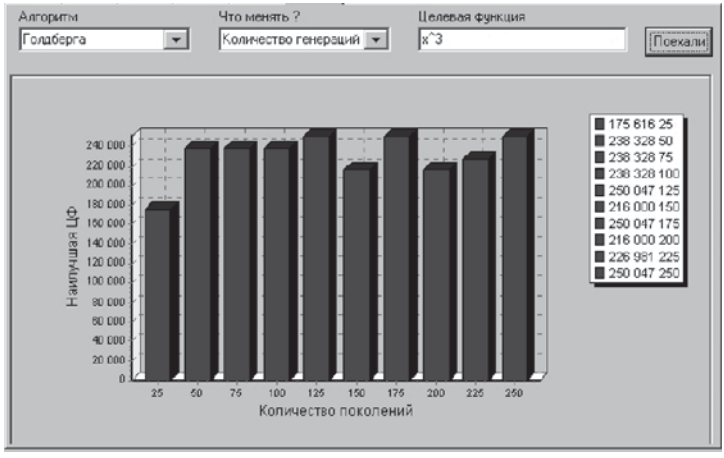


Рис. П.13. Пример окна вывода зависимостей результатов работы простого генетического алгоритма от заданных параметров (окончание)

Приложение 4. Методические указания к выполнению курсовой работы

1. Цель и задачи курсовой работы

Целью курсовой работы является изучение основных принципов эволюционной теории, механизмов передачи наследственной информации, ее сохранения и изменения в естественной среде, а также изучение и анализ существующих методов алгоритмов и архитектур генетического поиска, являющихся искусственными аналогами различных природных систем.

Для достижения данной цели студент должен решить следующие задачи:

1. Проанализировать полученное задание, выбрать необходимый программный инструментарий и на его основе разработать новый пакет прикладных программ, реализующий генетический алгоритм решения поставленной задачи с указанными в задании на курсовую работу функциями и свойствами.
2. Изучить существующие генетические операторы, методики и алгоритмы их выполнения, особенности различных генетических операторов, их достоинства и недостатки.
3. Выполнить программную реализацию стандартных схем выполнения генетических операторов, указанных в задании, а также, при необходимости, модифицировать либо разработать новые схемы выполнения стандартных генетических операторов в зависимости от характера решаемой задачи.

4. Изучить существующие архитектуры и схемы выполнения генетических алгоритмов, научиться создавать программные реализации этих схем, а также создавать новые модификации стандартных схем генетических алгоритмов;
5. Описать модифицированную схему генетического алгоритма, используемую в курсовой работе в виде структурной схемы алгоритма.
6. Провести, используя разработанный в ходе выполнения курсовой работы, комплекс программ, серии экспериментов для выявления характера влияния различных параметров генетического алгоритма на качество получаемых решений и построить графики зависимостей, полученных экспериментальным путем.
7. Изучить методику и порядок проведения статистической обработки полученных экспериментальных данных, а также нахождения теоретических и экспериментальных оценок параметров разработанного алгоритма.

2. Требования, предъявляемые к курсовой работе

Курсовая работа должна включать: пояснительную записку к курсовой работе (рекомендуемый объем пояснительной записки равен 20–25 листов машинописного текста формата А4); пакет прикладных программ, реализующий работу генетического алгоритма с заданными параметрами.

Структура и примерное содержание разделов пояснительной записки отражены в разделе 3 настоящего руководства.

Курсовые работы могут быть средней и повышенной сложности, а также исследовательские.

Курсовые работы средней сложности предусматривают решение простейших задач нахождения минимума — максимума функции на заданном числовом интервале с применением стандартных генетических операторов и схем простого генетического алгоритма.

В курсовых работах повышенной сложности студенты должны на основе полученных знаний разработать и применить к решению поставленных задач новые модификации стандартных генетических операторов и алгоритмов, учитывающие нюансы решаемой задачи.

Исследовательские курсовые работы включают в себя элементы научных исследований: сравнение и анализ существующих методов решения поставленной проблемы, разработку новых методик и схем выполнения генетических операторов, являющихся комбинацией существующих генетических операторов, а также новых архитектур и схем существующих алгоритмов, а в некоторых случаях и создание новых методик или схем алгоритмов.

3. Содержание пояснительной записки

Введение. Во введении должно быть изложено краткое содержание проделанной работы: указано, какая задача решалась, каким методом и на каком языке программирования, каковы возможности программ-

ного обеспечения, какие имеются ограничения, какая проведена обработка экспериментальных результатов работы программной реализации алгоритмов, какие качества алгоритма она выявила. Объем — около 1 страницы текста.

Постановка задачи. В этом разделе приводится математическое и неформальное описание исходных данных. В задании должна присутствовать постановка задачи, а также требования, допущения и ограничения на разработку алгоритма и программного обеспечения.

Анализ технического задания. Обоснование выбора генетического алгоритма как метода оптимизации для решения поставленной задачи. Следует показать, к какому классу относится решаемая задача.

Разработка генетического алгоритма

а) *Кодирование решения*. В разделе должен быть представлен способ кодирования решения задачи (конкретный пример структуры хромосомы и ее расшифровка, также с примером). Необходимо показать, как конкретный код соотносится с конкретным реальным решением.

б) *Оператор кроссинговера*. В разделе представляется оператор(ы) кроссинговера. Порядок выполнения оператора(ов) следует пояснить на конкретном примере.

в) *Оператор мутации*. В разделе представляются оператор(ы) мутации с соответствующими примерами.

г) *Другие генетические операторы*. Если в алгоритме используются дополнительные генетические операторы, они должны быть описаны в разделах с соответствующими названиями и показаны на примерах.

Структурная схема алгоритма. Приводится словесное описание и структурная схема генетического алгоритма, а также псевдокод алгоритма с комментариями и пояснениями.

Оценка сложности алгоритма. В разделе приводится расчет временной сложности как отдельных использованных в алгоритме операций, так и всего алгоритма в целом.

Экспериментальные результаты. В разделе указывается, какие параметры разработанного алгоритма исследуются, размер серии эксперимента, число экспериментов, а также полученные экспериментальные зависимости в виде таблиц и графиков.

Заключение. В заключении кратко описывается проделанная работа: какая задача решена, особенности разработанных алгоритмов и программного обеспечения. Необходимо также указать, при каких параметрах алгоритм работает эффективнее, задачи какой размерности решаются более успешно, подтвердилась ли теоретическая оценка временной сложности алгоритма. Объем — не более 1 страницы текста.

4. Типы задач для разрабатываемого генетическом алгоритме

4.1. Курсовые работы средней сложности. К задачам курсовых работ этой категории относятся стандартные оптимизационные задачи нахождения оптимума функции на некотором заданном числовом

интервале. Примеры задач такого рода приведены в описании к лабораторной работе № 5.

4.2. Курсовые работы повышенной сложности. К данной категории относятся стандартные задачи, успешно решаемые при помощи генетических алгоритмов, для которых требуется разработать новые комбинации или модифицированные схемы применения стандартных генетических операторов и алгоритмов. К числу таких задач можно отнести:

- алгоритмы проектирования печатных плат;
- алгоритмы проектирования больших и сверхбольших интегральных схем;
- задачу коммивояжера;
- задачу разбиения графа на части с оптимизацией разных критериев;
- задачу размещения вершин графа в узлы сетки с оптимизацией различных критериев;
- построение в графе, отображенном в узлы координатной решетки покрывающего дерева (дерева Штейнера) и т. д.

4.3. Исследовательские курсовые работы. В таких курсовых работах от студента требуется не только применить на практике полученные в ходе изучения теоретического курса знания, но и продемонстрировать умение и способности к научной и исследовательской работе. К числу задач данной категории относятся:

- задачи оптимизации;
- задачи проектирования;
- задачи на графах и гиперграфах;
- задачи распознавания образов;
- задачи искусственного интеллекта;
- задачи управления;
- задачи поддержки и принятия решений.

В частности:

- транспортная задача;
- задача выделения клик в графе;
- задача минимизации пересечений в графе;
- задача определения планарности в графе;
- задача определения изоморфизма графов;
- задача раскраски графа и т. д.

5. Варианты заданий

Индивидуальное задание на курсовую работу составляется и выдается преподавателем и представляет собой комбинацию различных типов генетических операторов.

Пречень возможных видов генетических операторов, которые могут использоваться при написании курсовой работы приводится ниже.

I. Кодирование хромосом
- A. Бинарное.
- B. Числовое.
- C. Векторное.

II. Стратегия создания начальной популяции
- A. Стратегия «одеяла».
- B. Стратегия «дробовика».
- C. Стратегия фокусировки.

III. Вид селекции
- A. Случайная.
- B. Селекция по заданной шкале.
- C. Элитная.
- D. Турнирная.
- E. Инбридинг.
- F. Гибридизация.

IV. Оператор кроссинговера
- A. Стандартный одноточечный.
- B. Стандартный двухточечный.
- C. Стандартный многоточечный.
- D. Универсальный.
- E. Упорядочивающий одноточечный.
- F. Упорядочивающий двухточечный.
- G. Частично соответствующий одноточечный.
- H. Частично соответствующий двухточечный.
- I. Циклический.
- J. Комплексный.
- K. «Жадный».
- L. Оператор кроссинговера на основе «Золотого сечения».
- M. Оператор кроссинговера на основе чисел Фибоначчи.

V. Операторы мутации и инверсии
- A. Простая.
- B. Точечная.
- C. Обмена.
- D. Обмена на основе «Золотого сечения».
- E. Обмена на основе чисел Фибоначчи.
- F. Инверсия.
- G. Дупликация.
- H. Транслокация.
- I. Транспозиция.

VI. Оператор отбора
- A. Пропорциональный.
- B. Элитный.
- C. Равновероятный.

VII. Схема эволюции

A. Микроэволюция.

B. Макроэволюция.

Пусть, например, задание на курсовую работу представляет собой следующую комбинацию номеров:

$$I\text{A}+II(\text{A, C})+III(\text{A, B})+IV(\text{A, C, G, L})+V(\text{B, H})+VI(\text{B})+VII(\text{A}).$$

Данная комбинация представляет вариант задания на курсовую работу, которое складывается из номеров определенных генетических операторов.

Разработать и реализовать программу генетического алгоритма, выполняющего следующие генетические операции:

*I*A — бинарное кодирование хромосом;

II(A, C) — варианты стратегии создания начальной популяции на основе стратегии «одеяла» и фокусировки;

III(A, B) — селекция случайная и по заданной шкале;

IV(A, C, G, L) — операторы кроссинговера — стандартные: одноточечный и многоточечный, частично соответствующий одноточечный и на основе чисел Фибоначчи;

V(B, H) — операторы мутации — точечная мутация и транслокация;

VI(B) — элитный отбор;

VII(A) — развитие популяции на основе схемы микроэволюции.

Заключение

В приложении 4 представлены примерные задания на курсовую работу, а также рекомендации по оформлению и оценке выполняемых курсовых работ. Приведенные задания носят рекомендательный характер и могут изменяться в зависимости от условий их выполнения.

Приложение 5. Примеры тестовых заданий по курсу

1. Основоположником генетики является:

- Дарвин
- Мендель
- Ламарк
- Вильсон

2. Впервые в науку термины «ген», «генотип», «фенотип», «аллель» и «локус», ввел:

- Мендель
- Иогансен
- де Фриз
- Морган

3. Пара одинаковых хромосом, несущих гены одинаковых признаков, называется:

- гомологичными хромосомами
- негомологичными хромосомами
- гомозиготными хромосомами
- генотипом

4. К факторам, не меняющим генетический состав популяции, относятся:

- кроссинговер
- «волны жизни»
- мутационный процесс
- отбор
- рекомбинация

5. Среди перечисленных положений к теории Моргана не имеют отношения следующие утверждения:

- наследственность и изменчивость организмов носит дискретный характер
- мутации отдельных генов приводят к изменению элементарных признаков организма
- чем меньше размер популяции, тем более вероятно, что в ней проявятся скрытые изменения
- гены находятся в хромосомах
- благодаря мутационному процессу состояние равновесия популяции непрерывно меняется

6. Из перечисленных видов мутаций не относятся к хромосомным:

- генные
- транслокации
- реципрокные
- геномные
- дицентрические

7. Среди перечисленных положений к мутационной теории де Фриза не имеют отношения следующие утверждения:

- мутации — это дискретные изменения наследственности
- мутации бывают вредными, полезными и нейтральными
- мутации возникают часто
- мутации не передаются по наследству
- мутации в природе спонтанны

8. Генетическая изменчивость может быть:

- направленной
- ненаправленной
- естественной
- популяционной
- мутационной

9. Основателем селекции как науки является:

- Дарвин
- Морган
- Ламарк
- Иогансен
- Мендель

10. Из перечисленных терминов видами отбора не являются:

- естественный
- неестественный
- сознательный
- бессознательный
- методический

11. Универсальными свойствами генетического материала являются:

- дискретность
- неопределенность
- непрерывность
- изменчивость
- линейность

12. Согласно теории Дарвина движущими силами эволюции являются:

- неопределенная изменчивость
- мутационный процесс
- борьба за существование
- естественный отбор
- внутривидовое сотрудничество

13. Многочисленная совокупность особей, в течении длительного периода населяющих определенный участок пространства, внутри которого осуществляется свободное скрещивание, называется:

- вид
- класс
- группа
- популяция
- сообщество

14. Для селекции используют следующие виды отбора организмов:

- бессознательный
- массовый
- целевой
- индивидуальный
- прогрессивный

15. Среди перечисленных ниже терминов моделями эволюции не являются:

- модель Дарвина
- модель Моргана
- модель де Фриза

- модель Ламарка
- модель Эйгена

16. Основоположником генетики является:

- Холланд
- Растригин
- Дэвис
- Дарвин

17. Генетические алгоритмы отличаются от других оптимизационных и поисковых процедур тем, что:

- используют целевую функцию, а не ее приращения
- дают точное решение задачи
- используют вероятностные правила
- имеют линейную временную сложность

18. При создании начального множества решений используются принципы:

- «одеяла»
- «матраса»
- «пулемета»
- «дробовика»

19. К числу операторов селекции относятся:

- селекция на основе рулетки
- селекция на основе гадания на картах
- чемпионская селекция
- турнирная селекция

20. Операторы кроссинговера могут быть:

- одноточечными
- многоточечными
- стоточечными
- бесконечноточечными

21. К операторам кроссинговера, использующим знания, относятся:

- циклический
- круизный
- специфический
- универсальный

22. Среди перечисленных положений к мутационной теории де Фриза не имеют отношения следующие утверждения:

- мутации — это дискретные изменения наследственности
- мутации бывают вредными, полезными и нейтральными
- мутации возникают часто
- мутации не передаются по наследству
- мутации в природе спонтанны

23. Оператор кроссинговера, делающий на каждом шаге преобразования локально-оптимальный выбор, называется:

- «грубым»
- «толстым»
- «проницательным»
- «жадным»

24. В генетических алгоритмах встречаются операторы:

- транслокации
- суперпозиции
- сегрегации
- дискриминации

25. Модель генетического алгоритма, предложенная Д. Холландом, называется:

- обычным генетическим алгоритмом
- простым генетическим алгоритмом
- универсальным генетическим алгоритмом
- базовым генетическим алгоритмом

26. Известными моделями генетических алгоритмов являются:

- модель Чамберса
- модель Девиса
- модель Клинтона
- модель Голдберга

27. Подмножество хромосом, содержащих знаки *, называется:

- стрингом
- вариантом
- хромосомой
- шаблоном

28. В зависимости от изменения размера популяции генетические алгоритмы делят на:

- динамические
- поколенческие
- стационарные
- подвижные

29. Число закрепленных позиций в шаблоне определяет его:

- длину
- ширину
- порядок
- класс

30. К методам одномерного поиска относятся:

- метод Монте-Карло
- метод градиента
- пассивный метод

- метод горизонта
- последовательный метод

31. В основе метода Фибоначчи лежит условие ..., где d_j — номер дискретной точки на интервале $[a, b]$:

- $d_j = d_{j+1} + d_{j+2}$
- $d_j = d_{j-1} + d_{j-2}$
- $d_j = d_{j-1} + d_{j+1}$
- $d_j = d_{j+2} - d_{j+1}$
- $d_j = d_{j+1} - d_{j-1}$

32. Сущность численных методов оптимизации заключается в построении последовательности векторов, удовлетворяющих условию:

- $f(x_{k-1}) + f(x_{k+1}) < f(x_k)$
- $f(x_{k+1}) > f(x_k)$
- $f(x_{k+1}) \leqslant f(x_k)$
- $f(x_{k+1}) < f(x_k)$
- $f(x_{k+1}) + f(x_k) < f(x_{k-1})$

33. В основе метода золотого сечения лежит условие ..., где d_j — номер дискретной точки на интервале $[a, b]$:

- $\frac{d_j}{d_{j-1}} = \frac{d_{j+2}}{d_{j+1}} = \text{const} = 1{,}618$
- $\frac{d_{j-1}}{d_{j+1}} = \frac{d_j}{d_{j-1}} = \text{const} = 1{,}618$
- $\frac{d_j}{d_{j-1}} = \frac{d_{j+2}}{d_{j+1}} = \text{const} = 1{,}018$
- $\frac{d_{j-1}}{d_j} = \frac{d_j}{d_{j+1}} = \text{const} = 1{,}618$
- $\frac{d_j}{d_{j-1}} = \frac{d_{j+2}}{d_{j+1}} = \text{const} = 0{,}618$

34. Основное уравнение градиентного поиска оптимума имеет вид:

- $x_{k+1} = x_k + h_k \operatorname{grad}(x_k - 1)$
- $x_{k+1} = x_{k-1} + h_{k-1} \operatorname{grad}(x_{k-1})$
- $x_{k+1} = x_{k-1} + h_k \operatorname{grad}(x_k)$
- $x_{k+1} = x_k + h_{k-1} \operatorname{grad}(x_{k-1})$
- $x_{k+1} = x_k + h_k \operatorname{grad}(x_k)$

35. Первым указал на фрактальную геометрию природы:

- Холланд
- Дарвин
- Мандельброт
- Мендель
- Кантор

36. Фрактальная размерность для множества Кантора, которое состоит из $N = 2n$ раздельных интервалов длины $\varepsilon = (1/3)n$, равна:
• 0,52 • 0,63 • 0,68 • 0,58 • 0,5

37. Значение нормализованной целевой функции рассчитывается по формуле:
• $f(P_i) = f_M(P_i) / \sum_{i=1}^{N_p} f_M(P_i)$
• $f(P_i) = 1/(1 + f_c(P_i))$
• $f_c(P_i) = f(\max) - f(P_i)$
• $f(P_i) = f_M(P_i) / \sum_{i=1}^{N_p} f_c(P_i)$
• $f(P_i) = f(\max) - fM(P_i)$

38. Основное отличие многоточечного оператора кроссинговера от оператора кроссинговера Фибоначчи заключается в том, что:
• в многоточечном ОК количество точек разреза ограничено
• в ОК Фибоначчи точки разреза являются фиксированными
• в многоточечном ОК точки разреза являются фиксированными
• в ОК Фибоначчи точки разреза соответствуют числам Фибоначчи
• ОК Фибоначчи может применяться только при длине хромосомы больше 8

39. В операторе инверсии «золотого сечения» первая точка разрыва находится на расстоянии ...
• $(L - 0{,}618L)0{,}618$ от любого края хромосомы
• $L - 0{,}618L)0{,}618$ от левого края хромосомы
• $(L - 0{,}618L)0{,}618$ от правого края хромосомы
• $0{,}618L$ от любого края хромосомы
• $0{,}618L$ от левого края хромосомы

40. Установленный порядок выполнения какой-либо деятельности или процесса называется:
• программой
• алгоритмом
• процедурой
• методикой

41. Процесс создания одной популяции решений и моделирования на этой популяции эволюционного поиска, называется:
• мини-эволюцией
• миди-эволюцией
• макро-эволюцией
• мета-эволюцией

42. Процедура, устанавливающая порядок обмена хромосом между популяциями, называется:
- мутацией
- транслокацией
- сегрегацией
- миграцией

43. Понятие кластерно-ориентированного генетического алгоритма первым ввел:
- Шафер
- Эшельман
- Парми
- Де Йонг
- Поппер

44. Групповой генетический алгоритм с направленной мутацией первым предложил:
- Шафер
- Эшельман
- Парми
- Де Йонг
- Абилов

45. Растения с доминантными признаками появляются во втором поколении потомства в соотношении:
- 5:1 • 1:1 • 2:1 • 3:1 • 4:1

46. Основной механизм эволюции по Дарвину есть:
- приобретенная наследственность
- естественный отбор
- скрещивание особей
- наследование приобретенных признаков
- мутация

47. Особь с двумя различными формами одного гена называется:
- гетеросексуальной
- гомосексуальной
- гетерозиготой
- гомозиготой
- гетеродином

48. Закон Харди–Вайнберга описывает:
- регламент генетической инженерии
- условия, при которых популяция достигает равновесия
- механизм естественного отбора
- размер популяции
- механизм свободного скрещивания в популяции

49. Нейтральные мутации:
- исчезают при генетическом дрейфе
- становятся полезными
- становятся вредными
- обуславливают изменчивость
- управляют генетическим дрейфом

50. Генетические алгоритмы — это:
- направление генетики
- вид отбора
- отрасль биологии
- метод оптимизации
- поисковая процедура

51. Процесс, посредством которого альтернативные решения с «лучшим» значением целевой функции имеют больше возможностей для воспроизводства потомков, называется:
- эволюцией
- отбором
- репродукцией
- селекцией
- мутацией

52. В процессе выбора пар хромосом для выполнения оператора кроссинговера могут использоваться следующие методы:
- метод инцеста
- метод «близкого родства»
- метод «око за око»
- метод на основе кода Хаффмана
- метод на основе кода Грея

53. Тесно связанные между собой части альтернативных решений, которые желательно сохранить в ходе выполнения генетических операторов есть:
- кирпичи
- фундамент
- арматура
- строительные блоки
- каркас

54. Для количественной оценки шаблонов в генетическом алгоритме используются следующие характеристики:
- порядок
- класс
- ранг
- определенная длина
- размер популяции

55. К методам последовательного поиска относятся:

- метод дихотомии
- метод градиента
- метод золотого сечения
- метод горизонта
- метод релаксации

56. Установите соответствие (метод поиска — его уравнение):

А) метод Фибоначчи

Б) метод золотого сечения

В) метод градиентного поиска

Г) метод наискорейшего подъема

Д) метод релаксации

1) $x_{k+1} = x_k + h_k \operatorname{grad}(x_k)$

2) $x_{k+1} = x_k + \alpha_k \nabla f(x_k)$

3) $d_j = d_{j+1} + d_{j+2}$

4) $\sum\limits_{j=1}^{m} \left(\dfrac{\partial f(x)}{\partial x_j} \right)^2 \leqslant \delta$

5) $\dfrac{d_{j-1}}{d_j} = \dfrac{d_j}{d_{j+1}} = \text{const} = 1{,}618$

57. Множество Кантора, промежуточное между точкой $d = 0$ и $d = 1$, называется:

- сечением
- интервалом
- отрезком
- фракталом

58. Оператор кроссинговера, в котором потомок формируется из r частей хромосом родителей, взятых по «диагонали», называется:

- многоточечным
- динамическим
- диагональным
- подвижным

59. Понятие проблемно-ориентированных генетических алгоритмов ввел:

- Шафер
- Эшельман
- Парми
- Де Йонг
- Поппер