Федеральное агентство по образованию

Нижегородский государственный университет им. Н.И. Лобачевского

Национальный проект «Образование»
Инновационная образовательная программа ННГУ. Образовательно-научный центр «Информационно-телекоммуникационные системы: физические основы и математическое обеспечение»

Д.И. Батищев, Е.А. Неймарк, Н.В. Старостин

Применение генетических алгоритмов к решению задач дискретной оптимизации

Учебно-методические материалы по программе повышения квалификации «Информационные технологии и компьютерное моделирование в прикладной математике»

Нижний Новгород

2007

Учебно-методические материалы подготовлены в рамках инновационной образовательной программы ННГУ: Образовательно-научный центр «Информационно-телекоммуникационные системы: физические основы и математическое обеспечение»

Батищев Д.И., Неймарк Е.А., Старостин Н.В. Применение генетических алгоритмов к решению задач дискретной оптимизации. Учебно-методический материал по программе повышения квалификации «Информационные технологии и компьютерное моделирование в прикладной математике». Нижний Новгород, 2007, 85 с.

Излагаются основы новой информационной технологии, позволяющей сводить классические задачи дискретной оптимизации, такие как комбинаторные задачи о ранце, коммивояжере, покрытии и разбиении, к задаче поиска на дискретном множестве кодировок. Рассматриваются основные принципы, типовые структуры и механизмы предлагаемого популяционно-генетического подхода к решению задач поиска с помощью генетических методов. Описаны основы генетического поиска и проанализированы математические модели генетических операторов кроссовера для разных типов представлений (кодировок). Приведены конкретные примеры, в которых большое внимание уделяется вычислительной реализации генетических методов.

Учебное пособие предназначено для преподавателей, аспирантов и специалистов, связанных с решением задач дискретной оптимизации. Также учебное пособие будет полезно студентам факультета вычислительной математики и кибернетики, изучающим курсы: «Методы и модели принятия решений» (общий курс по специальности «Прикладная информатика») и «Популяционно-генетический подход к решению экстремальных задач» (спецкурс по специальности «Прикладная математика и информатика»).

Оглавление

Предисловие

Проблемы принятия оптимальных проектных решений, возникающие в различных областях науки и техники, часто могут быть сформулированы как задачи дискретной оптимизации. Целью учебного пособия является изложение генетических методов, предназначенных для решения прикладных экстремальных комбинаторных задач переборного типа (задачи о ранце, задачи коммивояжера и д.р.), которые относятся к классу *NP*-трудных задач.

Генетический метод представляет собой скорее популяционно-генетический подход к решению задачи поиска, чем единый алгоритм решения дискретных оптимизационных задач. Таким образом, генетический метод образует класс алгоритмов поисковой оптимизации, основанных на математическом моделировании биологических механизмов и процессов в живой природе с помощью принципов популяционной генетики, которые позволяют находить оптимальные или близкие к ним (субоптимальные) решения.

Информационная технология решения задач дискретной оптимизации с помощью алгоритмов поиска, реализующих генетический метод, основывается на использовании аналогов с эволюционными процессами скрещивания, кроссовера, мутации и естественного отбора. Основные формализованные элементы информационной технологии подробно рассматриваются в пособии: типы представлений дискретных решений; схемы скрещивания; генетические операторы кроссовера и мутации; процедуры селекции; выбор параметров генетического метода, для обеспечения сходимости к субоптимальным решениям. Особое внимание в пособии уделено представлению в виде перестановок и представлениям, эквивалентным перестановочному (порядковое, угловое и представление смежности). Для каждого представления рассмотрен ряд генетических операторов кроссовера, которые формируют новые решения в виде перестановок на основе уже имеющихся перестановок.

Генетические методы не гарантируют обнаружения глобального оптимума за полиномиальное время, ибо только использование метода полного перебора позволяет найти решение глобальной оптимизации. Однако генетический алгоритм позволяет выбрать «достаточно хорошее»

решение за меньшее время, чем другие известные детерминированные или эвристические алгоритмы поисковой оптимизации.

Применение генетических методов для решения NP-трудных комбинаторных задач оптимизации полезно тогда, когда необходимый объем вычислительных затрат может оказаться большим, но скорость, с которой этот объем увеличивается при экспоненциальном росте «размерности» задачи дискретной оптимизации, часто может расти лишь линейно.

Нереально ожидать, что можно найти один, общий генетический алгоритм, который бы хорошо решал любую задачу дискретной оптимизации. Однако можно попытаться подобрать представления, операторы и параметры генетического алгоритма, адаптированного для конкретной задачи таким образом, чтобы они выполнялись оптимально или лучше, чем выбранные случайным образом.

Глава 1
Сведение комбинаторных задач дискретной оптимизации к задачам поиска

1.1. Постановки задач дискретной оптимизации

Решение большинства прикладных проблем, связанных с задачами выбора, управления и проектирования, заключается в построении *математической модели*, в которой отражается взаимосвязь наиболее важных и существенных для решаемой задачи характеристик *объекта исследования*. В качестве объекта исследования может выступать, например, техническое устройство, физический или технологический процесс, экономическая система и т.п. Подобные объекты исследования могут быть охарактеризованы совокупностью существенных свойств, которые могут быть объективно измерены. Количественная оценка существенных свойств описывается при помощи величин, называемых *параметрами*.

Выделяют *внешние параметры* $\bar{q} = (q_1, q_2, ..., q_n)$ – они характеризуют внешнюю по отношению к объекту среду, которая в той или иной степени оказывает влияние на функционирование объекта исследования. Как правило, подобные параметры задаются либо в форме констант, либо в форме функционалов от другой группы параметров – внутренних. *Внутренние параметры* описывают количественные характеристики составляющих элементов объекта исследования. Объединенную совокупность внешних и внутренних параметров будем называть *множеством входных параметров*.

Задача оптимизации возникает тогда, когда имеется набор (вектор) внутренних параметров $\bar{x} = (x_1, x_2, ..., x_n)$, который может принимать множество различных значений $\bar{x} \in D$. Такие параметры принято называть *управляющими переменными*. Собственно в определении конкретных значений управляющих переменных и состоит *акт принятия решения*.

Управляющие переменные оказывают влияние на свойства объекта исследования как единой системы. Подобное влияние может быть позитивным или негативным. Если в первом случае значимые для задачи

характеристики улучшаются, то во втором, соответственно, ухудшаются. Величины, характеризующие качественных показатели объекта исследования, будем называть *выходными параметрами* или *характеристиками* и обозначать $\bar{g} = (g_1, g_2, ..., g_m)$. Выходные параметры можно измерять, можно вычислять, но непосредственно изменять нельзя.

Управляющие переменные $\bar{x} \in D$ и характеристики $\bar{g}$ определяют существенные свойства объекта исследования. При этом $\bar{x} \in D$ играют роль независимых переменных, а $\bar{g}$ являются зависящими от них величинами $\bar{g} = \bar{g}(\bar{x})$. Множество D будем называть *областью поиска*, а любой вектор $\bar{x} \in D$ – *допустимым решением*.

Оптимизационные задачи формулируются как проблема выбора лучшего допустимого решения. Для определения понятия «лучшего» часто приходится вводить *критерий оптимальности* Q (или не один, а несколько критериев оптимальности) – количественный показатель, посредством которого осуществляется объективное измерение в некоторой числовой шкале Y одного наиболее важного для исходной задачи выходного параметра g_i. Под *измерением по шкале* Y понимается отображение Q, которое каждому решению $\bar{x} \in D$ ставит в соответствие числовую оценку $Q(\bar{x}) \in Y$ таким образом, чтобы отношения между числовыми значениями $Q(\bar{x})$ сохраняли бинарные *отношения предпочтения* между решениями:

1) $\bar{x}^1$ «лучше» $\bar{x}^2$ $(\bar{x}^1 P \bar{x}^2)$ тогда и только тогда, когда $Q(\bar{x}^1) > Q(\bar{x}^2)$;
2) $\bar{x}^1$ «не хуже» $\bar{x}^2$ $(\bar{x}^1 R \bar{x}^2)$ тогда и только тогда, когда $Q(\bar{x}^1) \geq Q(\bar{x}^2)$;
3) $\bar{x}^1$ «эквивалентно» $\bar{x}^2$ $(\bar{x}^1 I \bar{x}^2)$ тогда и только тогда, когда $Q(\bar{x}^1) = Q(\bar{x}^2)$.

(1.1)

Из соотношений (1.1) следует, что механизм выбора «лучшего» решения сводится к отбору тех и только тех решений, которые доставляют наибольшее значение критерию оптимальности Q в области поиска D:

$$Q^* = Q(\bar{x}^*) = \max_{\bar{x} \in D} Q(\bar{x}), \tag{1.2}$$

где $\bar{x}^*$ – *оптимальное решение*; $Q^* = Q(\bar{x}^*)$ – наибольшее значение критерия оптимальности среди всех значений критерия Q в области поиска D. Выражение (1.2) является математической записью модели принятия решения, называемой *экстремальной задачей однокритериального выбора*. В том случае, когда область поиска D состоит из счетного числа решений, принято говорить о задаче (1.2) как о *задаче дискретной оптимизации*.

Задача максимизации критерия является классической формой постановки задачи, к ней легко свести задачу, требующую минимизации критерия:

$$F(\bar{x}^*) = \min_{\bar{x} \in D} F(\bar{x}). \tag{1.3}$$

Переход к задаче максимизации можно осуществить, например, одним из трех способов:

$$Q(\bar{x}) = -F(\bar{x}), \tag{1.4}$$

$$Q(\bar{x}) = A - F(\bar{x}), \text{ где } A > 0 \text{ такое, что } Q(\bar{x}) < A, \bar{x} \in D, \tag{1.5}$$

$$Q(\bar{x}) = \frac{1}{F(\bar{x})}. \tag{1.6}$$

Максимизируемая (или минимизируемая) многопараметрическая функция, может быть как унимодальной, так и многоэкстремальной функцией. Независимо от вида функции $Q(\bar{x})$ оптимальное решение должно удовлетворять условию:

$$Q(\bar{x}^*) \geq Q(\bar{x}), \ \forall \bar{x} \in D. \tag{1.7}$$

В этом случае совокупность решений $\bar{x} \in D$, для которых неравенство (1.7) превращается в равенство, будем называть *глобальными решениями*.

В случае *унимодальной функции* (или *одноэкстремальной функции*) оптимальное решение единственно и достигается в точке *локального максимума*:

$$\begin{aligned} &Q(\bar{x}^*) \geq Q(\bar{x}), \ \forall \bar{x} \in d(\bar{x}^*, e), \\ &\text{где } d(\bar{x}^*, e) - e\text{-окрестность точки} \\ &\text{локального максимума } \bar{x}^* \in D. \end{aligned} \tag{1.8}$$

Многоэкстремальная функция предполагает несколько локальных максимумов. *Глобальный максимум* – наибольший из всех локальных максимумов. По аналогии формулируются понятия локальный и глобальный минимумы для задач минимизации (1.3).

В общем случае оптимальное решение переборной задачи может достигаться на подмножестве допустимых решений $\Omega \subseteq D$, удовлетворяющих условию:

$$Q^* = Q(\bar{x}), \ \forall \bar{x} \in \Omega. \tag{1.9}$$

Тогда в зависимости от постановки задачи однокритериального выбора требуется либо перечислить все решения из подмножества Ω, либо указать одно любое из решений этого подмножества.

Иногда в задачах дискретной оптимизации не удается выделить единственный наиболее важный выходной параметр, что приводит к *многокритериальной постановке оптимизационной задачи*:

$$\begin{cases} Q_1(\bar{x}^*) = \max\limits_{\bar{x} \in D} Q_1(\bar{x}), \\ Q_2(\bar{x}^*) = \max\limits_{\bar{x} \in D} Q_2(\bar{x}), \\ \quad \ldots \\ Q_m(\bar{x}^*) = \max\limits_{\bar{x} \in D} Q_m(\bar{x}). \end{cases} \tag{1.10}$$

Каждый из критериев задачи (1.10) Q_i где $i = \overline{1,m}$, называют *частным критерием* многокритериальной задачи оптимизации.

Из всего разнообразия подходов к решению многокритериальных задач [4,8] отметим два. Первый заключается в сведении задачи вида (1.10)

к задаче однокритериального выбора (1.2) методом свертки нескольких критериев в один обобщенный критерий. Другой подход предполагает введения новых отношений предпочтения, отличных от (1.1). Примером подобных отношении предпочтения могут служить отношения компромисса:

1) $\bar{x}^1$ «лучше» $\bar{x}^2$ тогда и только тогда, когда $Q_i(\bar{x}^1) \geq Q_i(\bar{x}^2)$ для любого $i = \overline{1,m}$ и существует по крайней мере один частный критерий, обращающий неравенство в строгое: $\exists j \in \{1,...m\} : Q_j(\bar{x}^1) > Q_j(\bar{x}^2)$;
2) $\bar{x}^1$ «не хуже» $\bar{x}^2$ тогда и только тогда, когда $Q_i(\bar{x}^1) \geq Q_i(\bar{x}^2)$ для любого $i = \overline{1,m}$; (1.11)
3) $\bar{x}^1$ «эквивалентно» $\bar{x}^2$ ($\bar{x}^1 I \bar{x}^2$) тогда и только тогда, когда $Q_i(\bar{x}^1) = Q_i(\bar{x}^2)$ для любого $i = \overline{1,m}$.

В случае отношений предпочтения (1.11) решением задачи будет уже не точка в пространстве поиска, но область из компромиссных решений.

1.2. Метод исчерпывающего перебора и понятие задачи переборного типа

В том случае, когда решение задачи (1.2) можно свести к анализу значений критерия оптимальности Q для конечного числа решений $\bar{x} \in D$ (1.3), говорят, что экстремальная задача однокритериального выбора относится к классу *экстремальных задач переборного типа* (или *переборных задач*).

$$\begin{cases} Q(\bar{x}^*) = \max\limits_{\bar{x} \in D} Q(\bar{x}), \\ 0 < |D| = N < \infty. \end{cases} \tag{1.12}$$

Здесь множество D – конечно, и количество возможных решений равно N. В силу этого формально все допустимые решения можно пронумеровать: $\bar{x}^1, \bar{x}^1, ...\bar{x}^N$, и поиск лучшего варианта из множества

допустимых решений сводится к полному перебору. Именно этим обстоятельством объясняется название переборных задач.

1.3. Оценка трудности задач дискретной оптимизации

Назовем алгоритм *эффективным*, если он способен найти решение задачи за время, полиномиально зависящее от объема входной информации, а задачи, имеющие такой алгоритм, – *полиномиально разрешимыми*. Являются ли задачи переборного типа полиномиально разрешимыми? Чтобы ответить на этот вопрос, оценим объем перебора.

Так как допустимое решение задачи (1.12) $\bar{x} \in D$ является n-мерным вектором $\bar{x} = (x_1, x_2, ..., x_n)$, каждый компонент которого может принимать одно из возможных значений из множества допустимых значений: $x_i \in D_i$, $|D_i| \geq 2$, $i = \overline{1,n}$, то можно рассчитать нижнюю оценку мощности множества допустимых решений:

$$N = \prod_{i=1}^{n} |D_i| \geq \prod_{i=1}^{n} 2 = 2^n. \qquad (1.13)$$

Таким образом, оценка объема вычислительных затрат на поиск оптимального решения задачи дискретной оптимизации показала, что затраты возрастают экспоненциально с ростом числа управляющих переменных.

Поскольку алгоритмы решения задач этого класса сводятся к перебору, то в общем случае задачи дискретной оптимизации не имеют эффективного алгоритма, находящего решение за время, полиномиально зависящее от размерности задачи.

1.4. Задача поиска и ее абстрактная модель

Поскольку множество допустимых решений конечно, и есть конечное количество значений, принимаемых каждой компонентой вектора решения, то произвольное допустимое решение можно закодировать целочисленным вектором $\bar{b} = (b_1, b_2, ..., b_n)$, где $b_i \in \{1,2,...|D_i|\}$, $i = \overline{1,n}$.

Здесь
D_i – множество допустимых значений i-ой компоненты вектора $\bar{x} \in D$.

$$(x_1, x_2, ..., x_n) \leftrightarrow (b_1, b_2, ...b_n), \;\; b_i \in \{1,2,...,D_i\}, \;\; i = \overline{1,n}. \qquad (1.14)$$

Рассмотрим некоторый конечный алфавит B, мощность алфавита обозначим через $|B|$. Для того, чтобы представить целочисленный вектор $b = (b_1, b_2 ... b_n)$ в алфавите B, необходимо определить максимальное число символов L, которое достаточно для представления любого значения b_i из области его допустимых значений D_i в алфавите B. Нетрудно видеть, что параметр символьной модели L должен удовлетворять неравенству:

$$D^+ < |B|^L, \;\; D^+ = \max_{i=\overline{1,n}} |D_i|. \qquad (1.15)$$

Запись произвольного целого неотрицательного числа $b_i \in [0, |B|^L)$ в алфавите B определяется соотношением:

$$b_i = \sum_{j=1}^{L} a_j |B|^{L-1}, \;\; a_j \in B, \;\; \forall j = \overline{1,L}, \qquad (1.16)$$

где L - необходимая длина кодирующего числа для b_i, $i = \overline{1,n}$.

Теорема 1.1. Длина кодирующего числа L для произвольного целого числа из области допустимых значений D мощности $|D|$ в алфавите B мощности $|B|$ определяется по формуле

$$L = \lceil \log_{|B|} |D| \rceil, \qquad (1.17)$$

где $\lceil Y \rceil$ – наибольшее целое число для положительного действительного числа Y.

Доказательство. Обозначим мощность множества допустимых решений D как $|D|$, количество возможных комбинаций из символов алфавита B длиной L равно $|B|^L$. Чтобы закодировать все допустимые решения при помощи строки длиной L в алфавите B, должно

выполняться условие $|D| < |B|^L$. Из этого следует, что $L = \lceil \log_{|B|} |D| \rceil$. Поскольку L может быть только целым числом, то $L = \lceil \log_{|B|} |D| \rceil > \log_{|B|} |D|$.

Теорема доказана.

Таким образом, с*имвольная модель экстремальной задачи переборного типа может быть представлена в виде множества бинарных строк, которые описывают конечное множество допустимых решений* $\bar{x}$ *из области поиска* D.

Фактически, чтобы перейти от задачи переборного типа к задаче поиска, необходимо закодировать допустимые решения, то есть построить отображение, функцию кодирования, переводящую множество допустимых решений в множество кодировок.

Функция кодирования является взаимно однозначным отображением множества допустимых значений D на множество кодировок $S = B^L$ длины L в алфавите B:

$$\Psi : D \to S,\ s^i = \Psi(x^i),\ s^i \in S,\ x^i \in D. \tag{1.18}$$

Поскольку функция кодирования устанавливает взаимно однозначное соответствие между D и S, можно построить *функцию декодирования*:

$$\Psi^{-1} : S \to D. \tag{1.19}$$

Теорема 1.2. Количество возможных функций кодирования при помощи строк длины L в алфавите B равно $\begin{pmatrix} |B|^L \\ |D| \end{pmatrix}$, где $|B|$ – мощность алфавита B, $|D|$ – мощность множества допустимых значений D.

Доказательство. Очевидно, что количество всевозможных кодировок длины L в алфавите B будет $|B|^L$. Для того, чтобы закодировать каждое из допустимых решений, нужно поставить ему в соответствие одну из кодировок, следовательно, каждая функция кодирования есть

сочетание вариантов из множества кодировок мощности $|B|^L$ по множеству допустимых решений мощности $|D|$. Очевидно, что количество всевозможных способов кодирования есть количество сочетаний из $|B|^L$ по $|D|$. Теорема доказана.

Кодирование может быть *ортогональным*, если каждой кодировке соответствует допустимое решение, и *неортогональным*, если кодирование предполагает существование таких кодировок, которые соответствуют недопустимым решениям исходной задачи дискретной оптимизации .

Для оценки кодировок определим *функцию цели* $g(s)$, измеряющую значение полезности кодировки в контексте исходной задачи оптимизации, как отображение $g: S \to R$. Функция цели для всех $s \in S$ является композицией двух функций $Q(z)$ и $z = \Psi^{-1}(s)$: $g(s) = Q(\Psi^{-1}(s))$. *Функция приспособленности* $m(s)$ является отображением $m: R \to R^+$ для всех кодировок $s \in S$:

$$m(s) = m(g(s)) = m(Q(\Psi^{-1}(s))) \geq 0. \qquad (1.20)$$

Задача поиска состоит в нахождении в конечном множестве кодировок S, состоящем из закодированных в алфавите B строк фиксированной длины L, такой строковой кодировки s^*, которая обеспечивает наибольшее значение функции приспособленности $m(s)$:

$$m(s^*) = \max_{s \in S} m(s) = \max_{s \in S} m(Q(\Psi^{-1}(s))). \qquad (1.21)$$

Приведем коммутационную диаграмму (см. рис. 1.1), связывающую задачу максимизации (1.2) с задачей поиска (1.21).

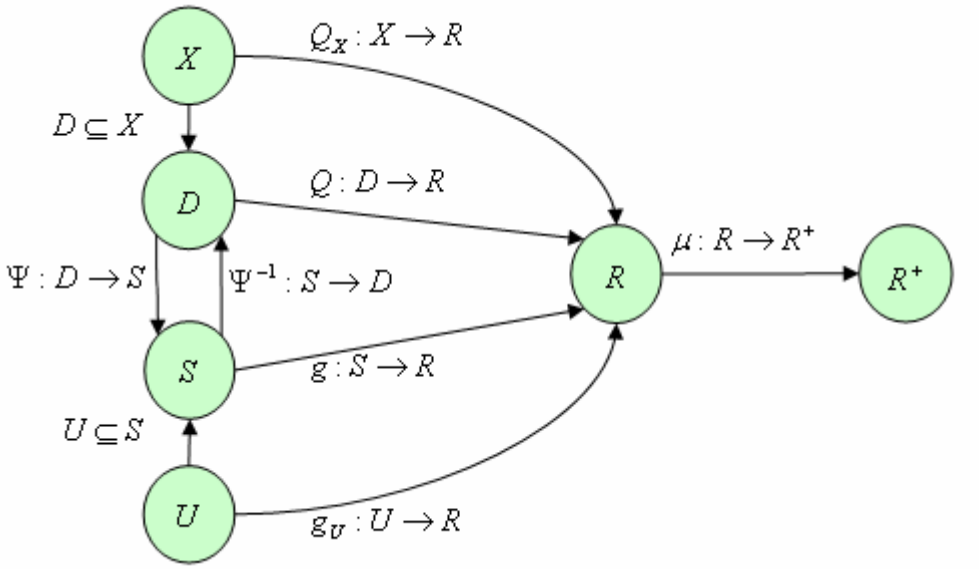

Рис. 1.1. Коммутационная диаграмма

На рисунке 1 приняты следующие обозначения:

- X – множество всевозможных решений (допустимых и недопустимых) задачи дискретной оптимизации (1.2);
- D – область допустимых решений задачи дискретной оптимизации (1.2);
- U – универсальное множество кодировок в виде всевозможных строк произвольной длины;
- S – пространство поиска, представляющее конечное множество строк фиксированной длины L;
- R – множество всех действительных чисел (R);
- R^+ – множество всех неотрицательных действительных чисел (R^+).

Отображениями множества D в S и наоборот являются функции кодирования и декодирования соответственно. Функции g и g_U – функции цели, определяющие полезности кодировок во множествах S и U. Они соответствуют задаче поиска. Функции Q и Q_X задают критерии оптимальности в пространствах D и X, они отражают задачу оптимизации.

Переход из R в R^+ относится к генетическому поиску и осуществляется посредством функции приспособленности m.

1.5. Бинарное представление дискретных решений с помощью двоичных чисел и кодов Грея

Классическим видом кодирования решений является кодирование решений в виде бинарных строк. Введем алфавит B^2, содержащий только два символа 0 и 1: $B^2 = \{0,1\}$.

Необходимая длина L кодирующего двоичного числа для представления в двоичном коде любого значения b_i из области его допустимых значений D_i можно вычислить по теореме 1. Запись произвольного целого неотрицательного числа $b_i \in [0,2^L]$ в двоичном виде определяется соотношением

$$b_i = \sum_{j=1}^{L} s_j^i \cdot 2^{L-1},\ s_j^i \in \{0,1\},\ j = \overline{1,L}. \qquad (1.22)$$

Тогда символьная запись целочисленного кода b_i для фиксированного значения управляющей переменной x_i в обычном двоичном коде запишется в виде следующей бинарной строки:

$$s_L(b_i):\ \ s_1^i, s_2^i ... s_L^i, \qquad (1.23)$$

где $s_j^i \in \{0,1\}$ $j = \overline{1,L}$ получены из соотношения (1.22).

Для представления допустимого решения $\bar{x} \in D$ экстремальной задачи в алфавите B^2 объединим символьные записи $s_L(b_i)$, описывающие все n компонент вектора $\bar{x}$, в виде линейной последовательности из бинарных комбинаций:

$$s(\bar{x}) = (s_L(b_1), s_L(b_2) ... s_L(b_n)). \qquad (1.24)$$

Записи (1.24) соответствует $(n \times L)$-битовая строка из двоичных символов (см. рис. 1.2).

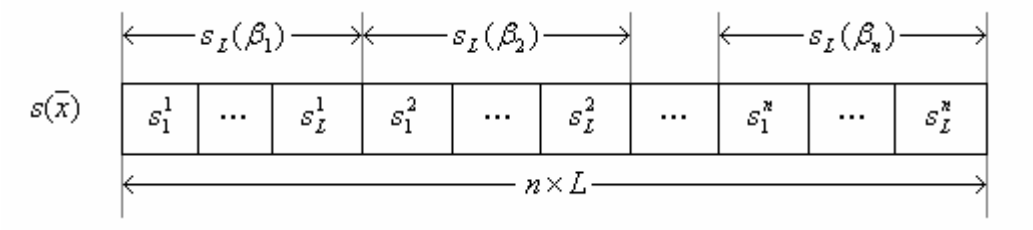

Рис. 2. Схема записи бинарной кодировки.

Для кодировок можно ввести понятие расстояния между бинарными кодировками $s', s'' \in S$, взяв в качестве меры *Хэммингово расстояние* между соответствующими строками. Расстояние $d(s', s'')$ между строками s' и s'' равно числу битов, содержащие различные значения.

Введем отображение $z : D \to \Omega \subset R^+$, которое каждому допустимому решению задачи (1.2) ставит в соответствие номер, который вычисляется из двоичной записи кодировки решения:

$$z(\bar{x}) = \Psi(\bar{x})_2. \qquad \textbf{(1.25)}$$

Бинарное представление решений $\bar{x}$ обладает тем недостатком, что в пространстве поиска S появляются так называемые *Хэммингловы обрывы*, когда в пространстве решений D подряд идущим целым числам (смежным числам) $z(\bar{x}')$, $z(\bar{x}'')$ соответствуют кодировки s', $s'' \in S$, $s' \neq s''$, которые имеют большое Хэммингово расстояние. Например, целые числа $z(\bar{x}')$=7 и $z(\bar{x}'')$=8 являются смежными, но их бинарные представления в обычном двоичном коде s' = 0111 и s'' = 1000 имеют Хэммингово расстояние, равное 4. Это приводит к тому, что кроссовер и мутация не могут легко преодолеть Хэммингловы обрывы. Например, для того чтобы изменить число $z(\bar{x}')$=7 на единицу, генетический алгоритм должен изменить одновременно все биты в кодировке s' на обратные значения.

Для того чтобы преодолеть указанные выше трудности, необходимо использовать в бинарном представлении коды, обладающие *свойством безразрывности кодирования*, состоящего в том, что изменение целого числа $z(\bar{x}') \in \Omega$ на единицу приводит к изменению в n-битовой строке смежного с ним целого числа $z(\bar{x}'')$ только в одном бите. Наиболее широко известными кодами, обладающими свойством безразрывности

кодирования, являются *рефлексивные двоичные коды Грея* (или просто *коды Грея*), в которых Хэммингово расстояние между кодировками любых двух смежных чисел $z(\bar{x}')$, $z(\bar{x}'') \in \Omega$ равно единице.

Пусть $s = (s_1, s_2, ..., s_n)$ – бинарное представление целого числа $z(\bar{x})$ в обычном коде, а $b = (b_1, b_2, ..., b_n)$ – в коде Грея. Тогда *преобразование обычного двоичного кода* s *в код Грея* можно задать с помощью следующего выражения:

$$b_k = \begin{cases} s_1, \text{ если } k = 1; \\ s_k \oplus s_{k-1}, \text{ если } k > 1, \ k = \overline{1,n}. \end{cases} \tag{1.26}$$

Здесь $\oplus$ – операция суммирования по модулю 2.

Например, для $z(\bar{x}')$=7 имеем s'=(0111), b'=(0100). Аналогично, для $z(\bar{x}'')$=8 имеем s''=(1000) , b''=(1100) . Хэммингово расстояние между кодировками b' и b'' равно единице.

Обратное преобразование кода Грея в обычный двоичный код s можно представить с помощью следующего выражения:

$$s_k = \bigoplus_{i=1}^{k} b_i, \quad k = \overline{1,n}, \tag{1.27}$$

где $\bigoplus_{i=1}^{k}$ – последовательное суммирование по модулю 2 всех битов представления в коде Грея от первого до k-ого бита.

Как показывает практика, на процесс генетического поиска и эффективность алгоритма в целом заметное влияние оказывает система кодирования решений задач дискретной оптимизации. В силу этого сложно переоценить важность этапа выбора символьной модели и системы кодирования решений исследуемой задачи. Рассмотрим другую, не двоичную схему представления дискретных решений, которую, как и две предыдущие схемы, можно назвать классической.

1.6. Небинарное (*n*-арное) представление дискретных решений

Существует ряд задач, для которых использование бинарного представления является нецелесообразным. Более естественным в этом случае будет использование n-арного кодирования, то есть использование алфавита B^n.

Одним из примеров может быть использование перестановки в качестве кодирующей строки в задачах, решение которых также является перестановкой. Каждая позиция в кодирующей строке может принимать одно из значений в пределах $[1,n]$.

Часто на такое представление накладывается ряд ограничений, которым должна удовлетворять кодировка. Для того, чтобы алгоритм поиска оперировал исключительно допустимыми кодировками, разрабатываются специальные операторы, гарантирующие получение допустимой кодировки. Они будут рассморены в следующих главах.

Таким образом, в задачах данного типа каждая управляемая переменная кодируется в некотором алфавите B, мощность которого равна n. В отличие от бинарного способа кодирования, этот способ называется *n-арным представлением*.

Примером n-арного кодирования может быть использование кодирования в виде перестановочного для *задачи коммивояжера*.

Решением классической задачи коммивояжера является перестановка без повторений из N натуральных чисел, представляющих порядковые номера вершин графа, длины N. Очевидно, что такой способ кодирования решений является ортогональным, однако применение операторов кроссовера и мутации к генотипам особей при перестановочном кодировании представляет ряд трудностей. При применении генетических операторов велика вероятность получения недопустимых кодировок, т.е. кодировок, которым не соответствует решение задачи.

Методы гарантированного получения допустимых кодировок при работе генетического алгоритма будут рассмотрены более подробно в параграфе 1.9.

1.7. Примеры экстремальных комбинаторных задач

Множество задач, имеющих одинаковую постановку и отличающихся только значениями параметров, будем называть *массовой задачей*. В дискретной оптимизации широко известные массовые задачи имеют названия, отражающие их интерпретацию.

1.7.1. Задача об одномерном ранце

Задача о ранце формулируется следующим образом: из заданного набора предметов $\bar{x}=\{x_1,...,x_n\}$, каждый из которых имеет вес w_i и ценность (стоимость) c_i, нужно выбрать несколько и наполнить ими ранец таким образом, чтобы суммарная выгода по выбранным предметам была наибольшей, а их общий вес не превосходил вместимость ранца P. Или формально:

$$\begin{cases} \max \sum_{i=1}^{n} x_i c_i; \\ \sum_{i=1}^{n} x_i w_i \le P; \\ x_i \in \{0,1\},\ i=\overline{1,n}. \end{cases} \tag{1.28}$$

$$x_i = \begin{cases} 1, & \text{если предмет с номером } i \text{ положен в ранец;} \\ 0, & \text{если предмет с номером } i \text{ не положен в ранец.} \end{cases} \tag{1.29}$$

Отметим, что $\sum_{i=1}^{n} w_i > P$, $c_i > 0$, $0 < w_i \le P$ для всех $i=\overline{1,n}$. Допустимым решением задачи о ранце является бинарный вектор $(x_1,...,x_n)$, удовлетворяющий ограничениям задачи. Очевидно, что допустимое решение всегда существует.

Задача легко кодируется при помощи бинарного представления. Число кодировок $N=2^n$, что и определяет вычислительную сложность задачи. Однако при таком кодировании не все получаемые кодировки будут допустимыми, поскольку соответствующие им решения могут нарушать ограничения на общий вес ранца. Предложенный способ кодирования

является неортогональным. Целесообразно исключать недопустимые кодировки из поиска.

1.7.2. Задача покрытия множества

Пусть задана булева матрица A размером $m \times n$, с каждым столбцом которой связано положительное значение стоимости $c_i > 0,\ i = \overline{1,n}$:

$$A = \begin{pmatrix} a_{11} & a_{12} & \mathbf{L} & a_{1n} \\ a_{21} & a_{22} & \mathbf{L} & a_{2n} \\ \mathbb{M} & \mathbb{M} & \mathbf{O} & \mathbb{M} \\ a_{m1} & a_{m2} & \mathbf{L} & a_{mn} \end{pmatrix} = (A_1, A_2, ..., A_n), \qquad (1.30)$$

где

$$a_{ij} = \begin{cases} 1, \text{если } i\text{-я строка матрицы } A \text{ содержит 1 на пересечении с } j\text{-м столбцом } A_j; \\ 0 \text{ в противном случае.} \end{cases}$$

Введем следующие обозначения:

- $I = \{1,2,...,m\}$ – множество индексов строк матрицы A;
- $J = \{1,2,...,n\}$ – множество индексов столбцов матрицы A;
- $c = \{c_1,...,c_n\}$ – вектор стоимости столбцов матрицы A;
- $P_j = \{i \in I : a_{ij} = 1\},\ j \in J$ – подмножество индексов строк, «покрываемых» столбцом A_j, т. е. имеющих 1 на пересечении с j-м столбцом;
- $R_i = \{j \in J : a_{ij} = 1\},\ i \in I$ – подмножество индексов столбцов, «покрывающих» i-ю строку, т. е. имеющих 1 на пересечении с i-й строкой.

Будем говорить, что подмножество индексов столбцов $J^{'}(J^{'} \subseteq J)$ является *покрытием матрицы* A, если соответствующие ему столбцы A_j, $j \in J^{'}$ «покрывают» все строки матрицы A:

$$\bigcup_{j \in J^{'}} P_j = I. \tag{1.31}$$

Покрытие $J^{*}(J^{*} \subseteq J)$, которому соответствует наименьшая суммарная стоимость столбцов A_j, $j \in J^{*}$ $\left(\sum_{j \in J^{*}} c_j \right)$, называется *покрытием минимальной стоимости* (или *минимальным покрытием матрицы* A).

Построение покрытия минимальной стоимости может быть сформулировано как задача целочисленного линейного программирования, если в качестве решения принять вектор $\bar{x} = (x_1, ..., x_n)$ с бинарными переменными $x_j \in \{0,1\}$, $j = \overline{1,n}$:

$$\begin{aligned} &\min_{\bar{x}} \left\{ \sum_{j=1}^{n} c_j \cdot x_j \right\}, \\ &\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = \overline{1,m}, \end{aligned} \tag{1.32}$$

$$x_j = \begin{cases} 1, \text{ если } j\text{-й столбец } A_j \text{ входит в покрытие;} \\ 0 \text{ в противном случае} \end{cases}$$

или в матричной форме:

$$\begin{cases} \min_{\bar{x}} \left(c \cdot x^T \right), \\ A x^T \geq \bar{1}, \\ x_j \in \{0,1\}, \quad j \in J, \end{cases} \tag{1.33}$$

где $\bar{1}$ – m-мерный вектор-столбец из единиц.

Как и в предыдущем случае, допустимым решением является бинарный вектор $\bar{x} = (x_1, ..., x_n)$, применяется кодировка в виде бинарной

строки длиной n. Число кодировок равно 2^n. Данный способ кодирования не является ортогональным.

1.7.3. Задача дихотомического разбиения графа

Пусть задан неориентированный граф $G(V,E,w)$ порядка n, где $V=\{v_1,v_2,...,v_n\}$ – множество вершин; E – множество ребер; $w: E \to R^+$ – отображение, определяющее вес каждого ребра.

Дихотомическим разбиением (V_1,V_2) будем называть такое разбиение графа $G(V,E,w)$ на два подграфа $G_1(V_1,E_1,w)$ и $G_2(V_2,E_2,w)$, что :

$$V_1 \subset V,\ V_2 \subset V,\ V_1 \neq \varnothing,\ V_2 \neq \varnothing; \tag{1.34}$$

$$V_1 \cup V_1 = V,\ V_1 \cap V_2 = \varnothing; \tag{1.35}$$

$$|V_1| = n_1,\ |V_2| = n - n_1, \tag{1.36}$$

где n_1 - целое положительное число $(0 < n_1 < n)$, которое задается как внешний параметр перед решением задачи разбиения.

Система требований (1.34)-(1.36), предъявленных к разбиению (V_1,V_2), определяет область поиска D.

В качестве критерия оптимальности Q, определяющего эффективность дихотомического разбиения (V_1,V_2), будем рассматривать вес разреза – сумму весов ребер, соединяющих вершины из разных подграфов:

$$Q(V_1,V_2) = \sum_{a \in V_1} \sum_{b \in V_2} w(a,b). \tag{1.37}$$

Тогда оптимальное дихотомическое разбиение (V_1^*,V_2^*) является оптимальным решением следующей экстремальной задачи однокритериального выбора:

$$Q^* = Q(V_1^*,V_2^*) = \min_{(V_1,V_2) \in D} \left(\sum_{a \in V_1} \sum_{b \in V_2} w(a,b) \right). \tag{1.38}$$

Задача относится к экстремальным задачам переборного типа, так как общее число допустимых решений равно $C_n^{n_1}$.

Решения можно кодировать в виде бинарных строк, значения 0 или 1 в i-ой позиции означают принадлежность i-ой вершины к соответствующему подмножеству вершин [1]. Очевидно, что данный способ кодирования допускает существование недопустимых кодировок, поэтому такое представление не является ортогональным.

1.7.4. Задача о назначениях

Пусть имеется n работ и n кандидатов на их выполнение, причем назначение j-й работы i-му кандидату требует затрат $c_{ij} > 0$. Задача состоит в нахождении такого распределения кандидатов по работам, чтобы минимизировать суммарные затраты. Причем каждый кандидат может быть назначен только на одну работу, и каждую работу может выполнять только один кандидат. Математически это можно записать:

$$\begin{cases} \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} c_{ij}x_{ij} \to \min; \\ \sum\limits_{i=1}^{n} x_{ij} = 1, \ \forall j = \overline{1,n}; \\ \sum\limits_{j=1}^{n} x_{ij} = 1, \ \forall i = \overline{1,n}; \\ x_{ij} \in \{0,1\}, \end{cases} \tag{1.39}$$

$$x_{ij} = \begin{cases} 1, \text{если кадидат } i \text{ назначается на работу } j; \\ 0, \text{в другом случае.} \end{cases}$$

Каждое решение можно закодировать перестановкой, в которой позиция обозначает номер кандидата, а значение в этой позиции – номер работы, назначенной кандидату. Очевидно, что число возможных назначений равно $N = n!$. Для задачи о назначениях n-арный способ кодирования будет обладать свойством ортогональности.

1.7.5. Задача коммивояжера

Задача коммивояжера является широко известной задачей оптимизации комбинаторного типа. Она формулируется следующим образом: дан полный взвешенный граф $G(X,V)$ порядка n, где $X=\{x_1,\ldots,x_n\}$ – множество вершин; $V\subseteq X\times X$ – множество ребер, в нем необходимо найти Гамильтонов цикл, имеющий наименьший суммарный вес входящих в него ребер.

Или формально :

$$\begin{cases} Q(x) = \sum\limits_{i-1}^{N}\sum\limits_{j=1}^{N} c_{ij}x_{ij} \to \min \\ \sum\limits_{i=1}^{N} x_{ij} = 1,\ \forall j = \overline{1,N} \\ \sum\limits_{j=1}^{N} x_{ij} = 1,\ \forall i = \overline{1,N} \\ x_{ij} \in \{0,1\} \end{cases}, \qquad (1.40)$$

где c_{ij} - вес ребра (i,j)

$$x_{ij} = \begin{cases} 1, \textit{в цикле есть переход из } i \textit{ в } j \\ 0, \textit{перехода из } i \textit{ в } j \textit{ нет} \end{cases}$$

Цикл в графе называется *Гамильтоновым циклом*, если он содержит все его ребра, причем каждое ребро один и только один раз.

Очевидно, что решением задачи является перестановка из n вершин, количество возможных перестановок равно $n!$, однако количество различных решений задачи с учетом направления обхода и сдвига начальной вершины будет $\frac{(n-1)!}{2}$. Задача не имеет алгоритма, позволяющего найти решение за приемлемое время, и решается в основном эвристическими методами.

Естественной формой кодирования для задачи коммивояжера является перестановка, то есть n-арное кодирование, где n – количество вершин графа. При этом способе кодирования в i-ой позиции кодирующей

строки стоит номер вершины, проходимой *i*-ой по порядку в цикле. Очевидно, что этот способ кодирования не является ортогональным.

1.8. Понятие окрестности решения для задач комбинаторного типа

Для допустимых решений задач комбинаторного типа определить понятие окрестности решения как для непрервыных функций, трудно или невозможно. Часто понятие близости вводится искусственно, основываясь на особенностях задачи.

Для Булева вектора $\bar{x} = (x_1, x_2 \dots x_n)$, являющегося решением задачи о ранце, определим в качестве окрестности множество всех векторов, получаемых из $\bar{x}$ заменой любого нуля на единицу или единицы на ноль (1.41). Иными словами окрестность бинарного вектора $\bar{x}$ составляют вектора, Хеммингово расстояние до которых равно единице.

$$\bar{x} \in d(\bar{x}^*), \ \textit{если} \ \forall i = 1,2 \dots j-1, j+1 \dots n \quad x_i = x_i^*, \ u \ \ x_j \neq x_j^* \qquad (1.41)$$

Определим в качестве расстояния между кодировками Хеммингово расстояние между ними (1.42).

$$r(\bar{x}, \bar{x}^*) = \sum_{i=1}^{n} x_i \oplus x_i^* \qquad (1.42)$$

Для задачи коммивояжера окрестностью (1.43) обхода $\bar{x} = (x_1, x_2 \dots x_n)$ будем называть все обходы, полученные из $\bar{x}$ перестановкой любых двух городов x_i и x_j, $i \neq j$.

$$\bar{x} \in d(\bar{x}^*), \ \textit{если} \ \forall i = 1, \dots j-1, j+1 \dots k-1, k+1 \dots n \quad x_i = x_i^*, \ u \ \ x_j = x_k^*, \ x_k = x_j^* \qquad (1.43)$$

Расстояние между двумя обходами для задачи коммивояжера определим как количество перестановок городов, необходимых для получения одного обхода из другого (1.44).

$$r(\bar{x},\bar{x}^*) = \sum_{i=1}^{n} salt(x_i, x_i^*)$$

$$salt(x_i, x_i^*) = \begin{cases} 0, \ salt(x_i, x_i^*), \ x_i = x_i^* \\ 1, \ x_i \neq x_i^* \ и \ \exists k, \ что \ (x_i = x_k^*) \ \& \ (x_k = x_i^*) \end{cases} \quad (1.44)$$

Очевидно, решения $x^1 \in D$ и $x^2 \in D$, близкие в области поиска D (согласно введенным понятиям близости), могут иметь сколь угодно далекие значения $Q(x^1)$ и $Q(x^2)$.

Область поиска D может быть несвязной и невыпуклой.

Кроме того, целевая функция может иметь несколько локальных оптимумов, отличных от глобального.

Процесс нахождения оптимального решения для задач комбинаторного типа сводится к полному перебору множества допустимых значений, очевидно, что объем вычислений зависит от количества допустимых решений. Если допустимое решение $\bar{x} \in D$ является вектором $\bar{x} = (x_1, x_2 \ldots x_n)$ каждая компонента которого $x_i, i = \overline{1,n}$ принимает значения из множества $D_i = (x_1^i, x_2^i \ldots x_{k_i}^i)$, $k_i \geq 2$, то объем вычислений можно оценить как $\prod_{i=1}^{n} | D_i | \geq \prod_{i=1}^{n} 2 = 2^n$, где $| D_i |$ - мощность множества D_i.

Учитывая приведенные особенности, алгоритмы решения задач этого класса сводятся к перебору, а из приведенной оценки следует, что вычислительные затраты возрастают экспоненциально с ростом числа управляемых переменных, то в общем случае задачи дискретной оптимизации не имеют эффективного алгоритма, находящего решение за время, полиномиально зависящее от размерности задачи.

1.9. Методы обработки ограничений

Исходные задачи оптимизации могут иметь ограничения, которые должны также быть отражены при генетическом поиске. Назовем особи, кодирующие недопустимые решения *нежизнеспособными*.

Например, задача о ранце имеет ограничение на суммарный вес предметов в ранце, в задаче коммивояжера каждая вершина графа должна быть представлена в решении один и только один раз.

Для отражения ограничений, существующих в исходной задаче, на пространство поиска используются различные методики. Во-первых это может быть *элиминация* нежизнеспособных особей, однако такой метод не всегда хорош, поскольку значительно сокращает генетическое разнообразие популяции и способствует преждевременной сходимости. Поэтому, к таким особям применяется *метод штрафных функций*, *метод восстановления генотипа*.

Идея использования штрафных функций состоит в значительном уменьшении приспособленности особей, не являющихся кодировками допустимых решений. Штрафные функции можно разделить на *постоянные* и *адаптивные*. Постоянная функция штрафа на все особи накладывает одинаковый штраф, адаптивная же накладывает штраф тем больше, чем «дальше» представляемое решение от допустимой области, например, квадратичная функция штрафа, приведенная в (1.46).

$f_{штраф} = const$	(1.45)
$f_{штраф} = (W_{особи} - W_{\max})^2$	(1.46)

После применения штрафа к функции приспособленности, некоторые особи, генотип которых является кодировкой недопустимого решения, могут иметь отрицательную или нулевую приспособленность.

Возможны два варианта работы с такими особями. Первый – это удаление особей с приспособленностью меньшей или равной нулю из популяции, но это значительно сокращает численность и генетическое разнообразие популяции. Второй вариант – масштабирование приспособленностей. В этом случае все особи сохраняются в популяции, но недопустимые особи имеют очень низкую, по сравнению с остальными особями, приспособленность. Одним из наиболее простых и часто используемых методов является *линейное динамическое масштабирование* (см. параграф 2.3.3).

Методы обработки ограничения с восстановлением генотипа преобразуют нежизнеспособное решение в жизнеспособное. При этом решения должны быть близки в смысле введенной метрики (1.42).

Например, для задачи о ранце может использоваться двойственный алгоритм Данцига. Алгоритм вычисляет значение относительной ценности

каждого предмета, положенного в рюкзак, эти значения упорядочиваются по возрастанию. Затем, из решения последовательно убираются предметы, имеющие наименьшую относительную ценность, до тех пор пока основное ограничение не будет удовлетворено**.**

Для задач с решениями – перестановками (задача коммивояжера, задачи теории расписания) при кодировании используются методы, гарантирующие допустимость решения, называемые также *декодерами*, и специальные операторы кроссовера и мутации .

Декодер модифицирует пространство поиска таким образом, что гарантирует получение допустимого решения. По сути, декодер может состоять из набора инструкций, позволяющих построить кодировку допустимого решения и перевести кодировку в допустимое решение.

Будем называть *чистым декодером* функцию кодирования, которая устанавливает взаимно однозначное соответствие между областью допустимых решений и их представлениями и не требует определения специальных операторов кроссовера и мутации, то есть использование классических операторов гарантирует декодирование в допустимое решение.

Поскольку решением задачи является перестановка, то естественным методом кодирования решения является перестановочное. Перестановочное представление – это строка, состоящая из n локусов, аллели каждого из генов являются натуральными числами в промежутке от 1 до n, значения генов повторяться не могут.

Перестановочное представление относится к методам ортогонального кодирования поскольку представление взаимно однозначно отражает решение.

Однако, применение классических видов кроссовера и мутации неприемлемо для перестановочного представления, поскольку не гарантируется получение допустимых потомков. Поэтому для него были разработаны специальные виды кроссовера (OX, CX, PMX и т.д.), применение которых гарантирует получение допустимых потомков.

Также для задачи коммивояжера можно использовать *представление смежности*. Здесь номер локуса соответствует номеру города, а значение гена в локусе – номер города, следующего за ним в обходе. Таким образом

это представление отражает смежность вершин в порядке следования, сохраняя этот порядок при кроссовере. В случае представления следования декодер переводит перестановку, являющуюся решением исходной задачи, в другую перестановку (кодировку), отражающую отношения следования между городами. При этом не любая строка, являющаяся перестановкой без повторений, в представлении следования может будет соответствовать допустимому решению (Гамильтоновому циклу), при декодировании она может распадаться на несколько малых циклов. Для этого представления существуют специальные операторы.

В качестве чистого декодера для задачи коммивояжера можно использовать *порядковое представление*. Это представление не требует специальных операторов и является ортогональным.

Глава 2
Основы генетического поиска

Как правило, задачи дискретной оптимизации, возникающие и требующие решения в процессе человеческой деятельности, характеризуются большим числом переменных, и, как следствие, большим пространством поиска, что не дает возможности перебрать все многообразие решений за «разумное» время. С другой стороны, на практике зачастую и не требуется нахождения глобального решения, достаточно найти «приемлемое» решение по заданному критерию (или набору критериев). Этими обстоятельствами объясняется повышенный интерес к комбинаторным методам оптимизации. В основе многих комбинаторных алгоритмов лежат эвристические схемы нахождения решений, которые эксплуатируют особенности объекта исследования. Однако и подобные методы не всегда удается использовать на практике, так как объект исследования может иметь очень непростую природу, и выявление закономерностей и особенностей его внутренней организации может оказаться нетривиальной задачей.

В связи с этим возникла проблема практической разрешимости задач дискретной оптимизации: найти эффективный или хотя бы достаточно простой в практически важных случаях алгоритм ее решения. Другой прием, который позволил сократить вычислительные затраты и соответственно увеличить размерность задач, заключается в ведении элементов рандомизации. Эволюционные методы представляют целый спектр алгоритмов, которые активно эксплуатируют данный прием. Как уже было отмечено, методы эволюционных вычислений не гарантируют обнаружения оптимального решения за полиномиальное время. Однако практический интерес к ним не ослабевает, а, наоборот, усиливается. Объяснить это можно тем обстоятельством, что эти методы позволяют исследовать и находить приемлемые решения таких задач, решение которых при помощи традиционных методов оказывается затруднительным, а в некоторых случаях и просто невозможным.

Идеи использования генетической модели передачи наследуемой биологической информации, хорошо исследованной генетиками, позволили создать семейство генетических алгоритмов. Сначала идеи генетики при решении задач оптимизации использовались для адаптации алгоритма

оптимизации. Начиная с работ Холланда (1975) и Де Янга, генетическими алгоритмами стали называть алгоритмы, моделирующие природную эволюцию в пространстве оптимизируемых параметров, а не в пространстве параметров алгоритма поиска.

2.1. Интерпретация экстремальной задачи поиска и операторов генетического алгоритма с помощью понятий популяционной генетики

Наименьшей неделимой единицей биологического вида, подверженной действию факторов эволюции, является *особь* a_k^t, (индекс k обозначает номер особи, а индекс t – некоторый момент времени эволюционного процесса). В качестве аналога особи a_k^t, в экстремальной задаче однокритериального выбора (1.2) примем произвольное допустимое решение $\bar{x} \in D$, которому присвоено имя a_k^t. Действительно, вектор управляющих переменных $\bar{x} = (x_1, x_2, ..., x_n)$ – это наименьшая неделимая единица, характеризующая в экстремальной задаче (1.2) внутренние параметры на каждом t-м шаге поиска оптимального решения, которые изменяют свои значения в процессе максимизации критерия оптимальности Q.

Как уже было отмечено, символьная модель экстремальной задачи переборного типа (1.2) может быть представлена в виде множества кодировок, будь то двоичных или n-арных строк, которые описывают конечное множество допустимых решений $\bar{x}$, принадлежащих области поиска D. Необходимо отметить, что выбор символьной модели исходной экстремальной задачи во многом определяет эффективность и качество применяемых генетических алгоритмов. Для каждого класса задач переборного типа должна строиться своя символьная модель, отражающая специфику и особенности решаемой задачи.

Для описания особей введем два типа *вариабельных признаков*, отражающих качественные и количественные различия между особями по степени их выраженности:

- *качественные признаки* – признаки, которые позволяют однозначно разделять совокупность особей на четко различимые группы;

- *количественные признаки* – признаки, проявляющие непрерывную изменчивость, в связи с чем степень их выраженности можно охарактеризовать числом.

Качественные признаки особи a_k^t определяются из символьной модели экстремальной задачи (1.2) как кодировка $s(\bar{x})$, соответствующая точке $\bar{x}$ с именем a_k^t и составляющие ее компоненты (бинарные или n-арные) $s_1(b_1), s_2(b_2),..., s_n(b_n)$. Приведем интерпретацию этих признаков в терминах хромосомной теории наследственности.

В качестве *гена* – единицы наследственного материала, ответственного за формирование альтернативных признаков особи, примем комбинацию $s_i(b_i)$, которая определяет фиксированное значение целочисленного кода управляющей переменной x_i. Каждая особь характеризуется n генами, а структуру строки $s(\bar{x}) = (s_1, s_2,..., s_n)$ можно интерпретировать *хромосомой*, содержащей n сцепленных между собой генов, которые следуют друг за другом в строго определенной последовательности. Хромосому особи a_k^t будем обозначать c_k^t, т.е.

$$c_k^t = c(a_k^t) = (c_1(a_k^t),\ c_2(a_k^t),...,\ c_n(a_k^t)) = s(\bar{x}) = (s_1, s_2,..., s_n). \qquad (2.1)$$

Согласно хромосомной теории наследственности передача генетической информации будет осуществляться через хромосомы от «родителей» к «потомкам».

Местоположение определенного гена в хромосоме называется *локусом*, а альтернативные формы одного и того же гена, расположенные в одинаковых локусах хромосомы, называются *аллелями*[1] (*аллелеформами*). В задаче поиска i-й локус соответствует i-й позиции в строковой кодировке $s(\bar{x})$, а аллели – это аналоги множества значений управляющих переменных.

Хромосомы, заполненные конкретными значениями называются *генотипами*. Генотип это строковая кодировка, состоящая из символов

[1] В генетике аллельные формы отвечают за проявления конкретных свойств

бинарного или n-арного алфавита. Кодировка $s(\bar{x}) \in S$ соответствует решению исходной задачи оптимизации $\bar{x} \in D$. Конечное множество всех возможных генотипов образует *генофонд* (*genofound*). В задаче поиска генофонд совпадает с пространством поиска S.

При взаимодействии особи с внешней средой ее генотип порождает совокупность *внешне наблюдаемых количественных признаков* (характеристик $g_1, g_2, ..., g_m$), называемых *фенотипом* $g(a_k^t)$. По аналогии с популяционной генетикой будем говорить, что реализуется «*принцип генного контроля*», когда множество генотипов взаимно однозначно отображается на множество фенотипов. С математической точки зрения этот принцип реализуется в силу взаимно однозначного отображения множеств D и S друг в друга.

Используя критерий оптимальности Q, можно получить оценку фенотипа, которая отображается на множество R^+. Подобную оценку можно интерпретировать как *приспособленность* $m(a_k^t)$ особи a_k^t. Чем больше значение *приспособленности*, тем лучше особь адаптирована к *внешней среде* Q. Таким образом, под приспособленностью особи $m(a_k^t)$ будем понимать склонность особи выживать и воспроизводиться в конкретной внешней среде, значение которой обычно запоминается для каждой особи.

ЗАМЕЧАНИЕ. Строго говоря, функция приспособленности m отображает множество кодировок S в множество неотрицательных действительных чисел R^+. Но так как для каждой a_k^t особи всегда однозначно определен свой хромосомный набор c_k^t, то в дальнейшем, будем считать запись $m(a_k^t)$ и запись $m(c_k^t)$ тождественными.

Следовательно, *цель эволюционного развития* особей сводится к определению такого генотипа, принадлежащего генофонду, который обеспечивает наибольшую приспособленность к внешней среде.

В качестве *ареала* – области, в пределах которой только и могут встречаться особи, участвующие в эволюционном процессе, будем рассматривать область поиска D. Совокупность особей $(a_1^t, a_2^t, ..., a_n^t)$, принадлежащих ареалу, образует популяцию P^t. Число n

характеризующее количество особей, которые образуют популяцию, будем называть *численностью популяции*.

Совокупность генотипов $(c_1^t, c_2^t, ..., c_n^t)$, соответствующих особям популяции P^t, образует *хромосомный набор* (*геном*) популяции P^t. При бинарном представлении хромосомы набор из n особей, каждая из которых характеризуется строкой фиксированной длины L из нулей и единиц, представляет из себя *булеву матрицу* (*boolean matrix*) размером $n \times L$.

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 популяций $P_i^t \subseteq P^t$, $i = \overline{1,k}$, включающих в свой состав те особи, которые имеют одинаковые или «достаточно близкие» формы тех или иных качественных или/и количественных признаков.

В качестве количественного признака можно использовать степень приспособленности особей $m(a_k^t)$. В том случае, когда для дифференциации особей a_k^t используется качественный признак, например, генотип $s(\bar{x})$. В качестве меры «близости» особей a_i^t и a_j^t можно использовать число несовпадающих по своим значениям геном в строках c_i^t и c_j^t. В случае бинарного представления такую оценку называют *Хэмминговым расстоянием*, и она рассчитывается по формуле:

$$d\left(c_i^t, c_j^t\right) = \sum_{l=1}^{n} c_l(a_i^t) \oplus c_l(a_j^t), \qquad (2.2)$$

где $\oplus$ – операция суммирования по модулю 2.

Будем считать, что во времени популяции P^t состоят из дискретных, не перекрывающихся между собой *поколений* – групп особей, одинаково отдаленных в родственном отношении от общих предков, т.е. каждое последующее поколение P^{t+1} является совокупностью из n особей, которые отбираются только из особей предыдущего t-го поколения. Будем отождествлять номер поколения (верхний индекс t в обозначениях особи

a_k^t и популяции P^t) с моментом времени $t = 0,1,...,T$, где T – *жизненный цикл* популяции, определяющий период ее эволюции.

В дальнейшем *эволюцию популяции* P^t будем понимать в ограниченном смысле как чередование поколений, в процессе которого особи изменяют свои вариабельные признаки таким образом, чтобы каждая следующая популяция проявляла лучшую степень приспособленности к внешней среде, например, в смысле обеспечения наибольшего значения *средней степени приспособленности* по популяции P^t:

$$m_{cp}(t) = \frac{1}{n}\sum_{i=1}^{n} m(a_i^t). \qquad (2.3)$$

Совокупность из n генотипов всех особей $(a_1^t, a_2^t, ..., a_n^t)$, составляющих популяцию P^t, образует хромосомный набор $(c_1^t, c_2^t, ..., c_n^t)$, который полностью содержит в себе генетическую информацию о популяции P^t в целом. Наличие изменчивости хромосомного набора от поколения к поколению является необходимым условием эволюции популяции P^t на генетическом уровне.

По хромосомному набору $(c_1^t, c_2^t, ..., c_n^t)$ популяции P^t можно определить *частоту генотипа* (или *генотипическое разнообразие*) $P^t(c)$ как долю особей, имеющих одну и ту же форму генотипа c в рассматриваемой популяции P^t. Популяцию P^t, в которой хромосомный набор состоит из n одинаковых генотипов, будем называть полностью сходящейся популяцией (*totally converged population*).

Как уже отмечалось, в процессе эволюции хромосомный набор популяции меняется, что обеспечивается *природными механизмами* наследственности, изменчивости и естественного отбора. Генетический алгоритм, моделируя природные механизмы, обеспечивает эволюцию популяции от поколения к поколению. *Целью генетического поиска* является поиск особи с наибольшей степенью приспособленности. Таким образом, природным аналогом процесса генетического поиска является процесс естественной эволюции.

Приведем интерпретацию основных составляющих генетического метода в терминах популяционной генетики.

Пусть популяция $P^t = (a_1^t, a_2^t, ..., a_n^t)$ является *Менделевой популяцией*, в которой любые две особи $a_i^t, a_j^t \in P^t$, объединенные в одну «родительскую» пару (a_i^t, a_j^t), могут произвести «потомство», генотипы которых будут содержать информацию, унаследованную от «родителей». Способ подбора особей в пары родителей будем называть *системой скрещивания* (*breeding*). В генетическом алгоритме ответственность за реализацию системы скрещивания лежит на *операторе скрещивания* – аналог механизма формирования «брачных» пар в процессе естественной эволюции.

Пусть особи $a_i^t, a_j^t \in P^t$ с различающимися генотипами являются «родительской» парой, образованной при помощи некоторой системы скрещивания. Процесс воспроизводства новых особей – «потомков» в природе построен на принципе наследственности вариабельных признаков от «родителей» к «потомкам». Реализуется этот принцип *механизмом рекомбинации генов*. По своей сути рекомбинация генов ведет к появлению новых сочетаний «родительских» генов в генотипах потомков. Подобный механизм в генетическом алгоритме реализован в форме *операторов размножения*. Оператор размножения называют *кроссовером*, если он поострен на рекомбинации генов, т.е. новые особи формируются методом взаимного обмена участками «родительских» хромосом. Особей текущей популяции P^t будем называть *родителями*. Особей, полученных в результате работы операторов размножения, будем называть *потомками*.

Фактически оператор кроссовера не вносит новой генетической информации в популяцию. Источником генетической изменчивости являются *мутации* – изменения аллелей в одном или нескольких генах хромосом «потомков» в результате появления в них новых аллелей, не содержащихся ранее в соответствующих генах. Для обеспечения свойства изменчивости в генетическом алгоритме присутствует *оператор мутации*. Особей, полученных в результате мутации, будем называть *мутантами*. Мутации являются стохастическими в том смысле, что не зависят ни от генетических кодов особи, ни от значений ее фенотипа и функции приспособленности.

Как уже отмечалось, естественный отбор – это процесс формирования популяции P^{t+1}, который способствует «выживанию» более приспособленных к внешней среде особей и «элиминации» тех особей, которые имеют пониженную приспособленность к внешней среде. Оператор, формирующий популяцию следующего поколения и реализующий принцип естественного отбора в генетическом алгоритме, называется *оператором отбора*.

Рассмотрим обобщенную структуру и каждый из операторов генетического алгоритма.

2.2. Обобщенная структура генетического алгоритма

Прежде чем привести обобщенную структуру генетического алгоритма, введем ряд определений.

Репродукционно-популяционный алгоритм поиска – это алгоритм поисковой оптимизации, который начинается с начальной популяции P^0 (совокупности кодировок $(c_1^0, c_2^0, ..., c_n^0)$) и итеративно выполняет следующий цикл операций:

1) *Оценивание* – вычисление значений функции приспособленности для любой кодировки $c \in S$.

2) *Селекция* – выбор из популяции P^t, $t = 0,1,...$ репродукционного множества R^t (подмножества кодировок $R^t \subseteq P^t$).

3) *Воспроизводство* (*размножение*) – генерация из репродукционного множества R^t новых кодировок с помощью комбинаций следующих операций:

   а) *копирование* – создание тождественных копий некоторых или всех кодировок из R^t;

   б) *скрещивание* – конструирование новых кодировок путем сцепления подстрок тех кодировок, которые выбираются путем копирования из R^t;

   в) *мутация* – конструирование новых кодировок путем подстановки символов из алфавита B в выбранные позиции одной из кодировок $c \in R^t$.

4) *Замена* – формирование на очередном шаге (поколении) новой популяции P^{t+1} путем замены некоторых или всех кодировок $c \in P^t$.

Канонические генетические алгоритмы – это подкласс репродукционно-популяционных алгоритмов поиска, в которых основные операторы имитируют механизмы естественного отбора и популяционной генетики, удовлетворяющий следующим требованиям:

- все кодировки $c \in S$ имеют одну и ту же длину L;
- каждая популяция P^t, $t = 0,1,...$ имеет постоянную численность n;
- мощность репродукционного множества $|R^t|=2$; элементы $c', c'' \in R^t$, называемые *родителями*, выбираются из популяции P^t случайным образом с вероятностями, пропорциональными значениям функции приспособленности;
- скрещивание осуществляется с помощью *одноточечного кроссовера*, когда символы $(c_1, c_2,..., c_i)$ в новой кодировке c, называемой «потомком», являются символами кодировки $c' \in R^t$, а символы $(c_{i+1}, c_{i+2},..., c_L)$ переходят от кодировки $c'' \in R^t$; индекс i, называемый *точкой кроссовера*, выбирается случайным образом с равной вероятностью из интервала [1, L-1];
- мутации являются стохастическими операциями и обычно не зависят от значений функции приспособленности;
- замена сводится к исключению из популяции P^t особи с наименьшим значением функции приспособленности и включению в нее одной из новых кодировок с наибольшим значением функции приспособленности.

Приведем отличия генетических алгоритмов (ГА) от традиционных методов оптимизации:

1) ГА манипулируют непосредственно кодировками $c \in S$, а не самими решениями $\bar{x} \in D$ исходной задачи дискретной оптимизации.
2) ГА работают не с одной кодировкой $c \in S$, а одновременно с совокупностью кодировок $(c^1, c^2,..., c^n)$ из пространства поиска S,

называемой *популяцией* P^t численностью n, которая формируется заново для каждого последующего поколения.

3) ГА являются *методами нулевого порядка* («*слепого*» *поиска*), использующими в процессе поиска только информацию об оценке кодировок $c \in S$ с помощью функции приспособленности.

4) ГА основываются на моделировании биологических механизмов эволюции и популяционной генетики в живой природе, а не на математических свойствах функции приспособленности.

5) ГА являются *стохастическими алгоритмами*, основные операторы которых (селекция, скрещивание, мутация, замена) основываются на вероятностных схемах.

6) ГА являются *робастными методами* по отношению к виду оптимизируемого критерия оптимальности с точки зрения вычислительных затрат, требуемых на поиск.

Обобщенная структура генетического алгоритма может быть представлена следующим образом.

1. **Инициализация начальной популяции P^0 численностью n.**

   1.1. Установить номер текущего поколения $t := 0$.

   1.2. Сгенерировать случайным образом хромосомный набор из n строковых кодировок фиксированной длины L, в котором Хэммингово расстояние между любой парой кодировок не равно нулю.

   1.3. Оценить каждую строку из хромосомного набора с помощью функции приспособленности.

2. **Воспроизводство потомков с наследственными генетическими свойствами родителей.**

   2.1. Выбрать случайным образом из текущей популяции P^t согласно *схеме скрещивания* кодировки двух родителей, образующих «брачную пару».

   2.2. Сгенерировать при помощи оператора кроссовера для выбранной «брачной пары» с вероятностью p_c одну или несколько кодировок потомков, которые наследуют генетические свойства родителей.

2.3. Оценить каждую кодировку потомков с помощью функции приспособленности.

2.4. Повторять все операции с п. 2.1 до тех пор, пока не будет рассмотрено заданное число «брачных пар» N_c.

3. Создание мутантов с генетическими свойствами, отличающимися от свойств родителей.

3.1. Выбрать случайным образом из числа потомков и/или родителей кодировку, наследующую генетические свойства одного или обоих родителей.

3.2. Сгенерировать при помощи оператора мутации для выбранной кодировки с вероятностью p_m кодировку-мутанта, обеспечивающую изменчивость генетических свойств родителей.

3.3. Оценить кодировку мутанта с помощью функции приспособленности.

3.4. Повторять все операции с п. 3.1 до тех пор, пока не будет получено заданное число мутантов N_m.

4. Замена текущей популяции P^t новой популяцией P^{t+1}.

4.1. Выбрать стратегию формирования популяции P^{t+1}.

4.2. Сформировать из родителей и/или «детей» (потомков и мутантов) репродукционное множество кодировок, различающихся между собой по Хэммингову расстоянию.

4.3. Скопировать при помощи оператора селекции из репродукционного множества кодировки, реализующие стратегию формирования популяции P^{t+1}.

5. Условие выхода из итерационного цикла. Сменить номер текущего поколения $t := t+1$ и повторить все операции с п.2, если *условие окончания генетического поиска* не выполнено (например, эволюция популяции P^t считается законченной, если она исчерпала свой жизненный цикл T, т. е. если $t > T$).

Параметрами генетического алгоритма являются:

- n – численность популяции P^t, $t = 0,1,...$;
- N_c – число «брачных пар»;
- N_m – число «мутантов»;
- p_c – вероятность кроссовера;
- p_m – вероятность мутации;
- T – число поколений, в течение которых осуществляется генетический поиск.

Остановимся более подробно на основных операторах генетического алгоритма.

2.3. Операторы генетического алгоритма, не зависящие от типа представления

Символьная модель задачи дискретной оптимизации (представление решений) во многом определяет операторы генетического алгоритма. Однако в обобщенной структуре генетического алгоритма можно выделить ряд операторов, которые не зависят от типа представления, кроме того, в них никак не проявляется ни специфика задачи, ни особенности системы кодирования. К таким операторам можно отнести оператор скрещивания, замены, селекции и масштабирования функции приспособленности.

2.3.1. Оператор скрещивания

В генетическом алгоритме ответственность за формирования «родительских» пар особей лежит на *операторе скрещивания*. Данный оператор может использовать разные стратегии в подборе пар. Подобные стратегии принято называть *системами скрещивания*.

В качестве способа подбора пары родительских кодировок для дальнейшего размножения может использоваться одна из следующих *систем скрещивания*.

Панмиксия (*panmixia*)– система скрещивания, в которой любые две особи a_i^t и a_j^t имеют одинаковую возможность образовать случайным образом «брачную пару». Если при скрещивании для особей с

одинаковыми генотипами исключается возможность оказаться в одной паре, то имеет место *панмиксия генотипов*.

Инбридинг (*inbreeding*) – система скрещивания, в которой при образовании «брачной пары» предпочтение отдается особям с генетически похожим кодировкам c_i^t, c_j^t, удовлетворяющим условию:

$$0 < d\left(c_i^t, c_j^t\right) \le d^+,$$
$$\text{где } d\left(c_i^t, c_j^t\right) = \sum_{l=1}^{n} c_l(a_i^t) \oplus c_l(a_j^t), \tag{2.4}$$

d^+ – параметр инбридинга, регулирующий степень родства особей в парах.

Аутбридинг (*autobreeding*) или кроссбридинг – система скрещивания, в которой при образовании «брачной пары» предпочтение отдается особям с генетически не похожими кодировкам c_i^t, c_j^t, удовлетворяющим условию:

$$d^- \le d\left(c_i^t, c_j^t\right) \le L,$$
$$\text{где } d\left(c_i^t, c_j^t\right) = \sum_{l=1}^{n} c_l(a_i^t) \oplus c_l(a_j^t), \tag{2.5}$$

d^- – параметр аутбридинга, регулирующий степень непохожести особей в парах,
L – длина кодировок.

Ассоциативное скрещивание (*associative mating*) – система скрещивания, в которой при образовании «брачной пары» особи выбираются только на основании информации об их значениях функции приспособленности (или соответствующих им фенотипов).

Положительное ассоциативное скрещивание (*positive associative mating*) – система скрещивания, в которой при образовании «брачной пары»

предпочтение отдается особям с близким значениями оценки приспособленности[1]:

$$\left| m(a_i^t) - m(a_j^t) \right| \le dm ,$$

где dm – допустимая разность между значениями функций приспособленности особей a_i^t и a_j^t. (2.6)

Отрицательное ассоциативное скрещивание – система скрещивания, в которой при образовании «брачной пары» предпочтение отдается особям с сильно различающимся между собой значениями функции приспособленности[2]:

$$\left| m(a_i^t) - m(a_j^t) \right| \ge dm ,$$

где dm – параметр, регулирующий степень различия приспособленности особей a_i^t и a_j^t. (2.7)

Селективное скрещивание – система скрещивания, в которой при образовании «брачной пары» осуществляется предварительное отстранение некоторых кодировок от участия в этом процессе (например, по условию, что в парах участвуют только те особи, для которых $m(a) > m^-$), а оставшиеся особи выбираются в «брачную пару» по одной из определенных выше систем скрещивания.

2.3.2. Стратегии формирования популяции при переходе от одного поколения к другому

Важным компонентом генетического алгоритма является *стратегия замены* текущей популяции P^t новой популяцией P^{t+1} следующего $(t+1)$-го поколения. В качестве параметра управления стратегиями замены

[1] В некоторых источниках под положительным ассоциативным скрещиванием понимают такую систему, которая при образовании брачных пар отдает предпочтение наиболее приспособленным особям

P^t на P^{t+1} используется параметр $G(0 < G \le 1)$, называемый *интервалом перекрытия поколений*, который определяет ту часть g особей из популяции $P^t = \left(a_1^t, ..., a_n^t\right)$, которые должны быть новыми в популяции P^{t+1}:

$$g = G \cdot v, \tag{2.8}$$

где n – численность популяций P^t и P^{t+1}; $(v - g)$ – число особей, копируемых из популяции P^t в популяцию P^{t+1}; g – число особей, копируемых вновь из потомков и мутантов, полученных в t-ом поколении в популяцию P^{t+1}.

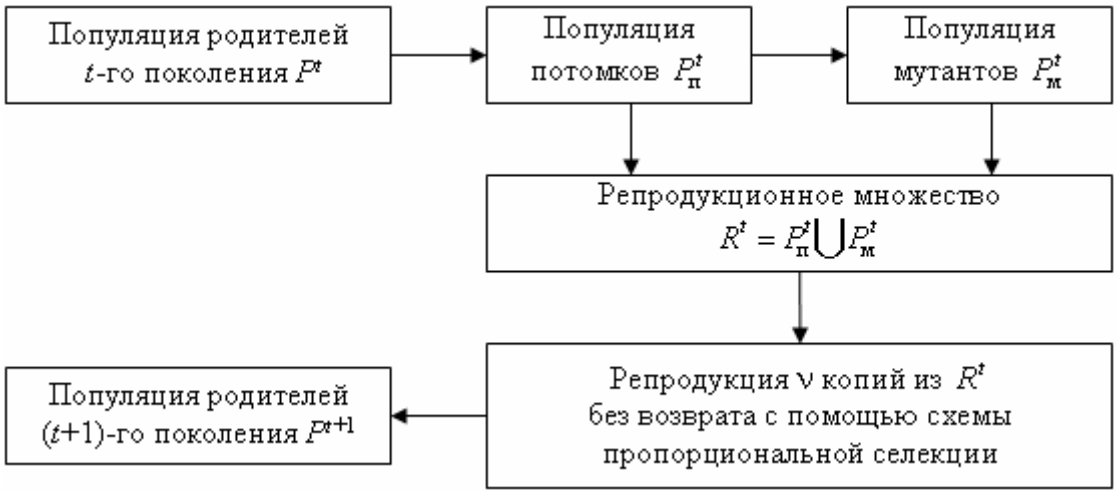

Рис. 2.1. Поколенческая репродукция $G = 1$

Если $G = 1$, то все n особей должны быть заменены новыми структурами, копируемыми только из совокупности потомков и мутантов, воспроизведенных в t-м поколении с помощью операторов кроссовера и мутации, соответственно (см. рис. 2.1). Такая стратегия формирования популяции P^{t+1}, называемая *поколенческой стратегией (generational strategies)*, позволяет заменять сразу всю популяцию P^t полностью новой популяцией P^{t+1}, которая состоит только из копий потомков и мутантов. Согласно выражению (2.8) все n особей популяции P^{t+1} отличаются от особей популяции P^t. Это позволяет говорить, что популяции P^t и P^{t+1}, численность которых постоянна и равна n, являются *неперекрывающимися популяциями* (*nonoverlapping populations*).

Недостатком поколенческой стратегии является то, что время жизни «родителей» из популяции P^t равно только одному поколению. В результате многие из «хороших» решений, составляющих популяцию P^t, могут не воспроизвестись в дальнейшем вообще или их гены могут быть потеряны в процессе генетического поиска.

Этот недостаток может быть частично исправлен с помощью *элитной поколенческой схемы* (*elitist generational strategies*), которая гарантирует сохранение единственной лучшей особи, полученной в *t*-м поколении (см. рис. 2.2).

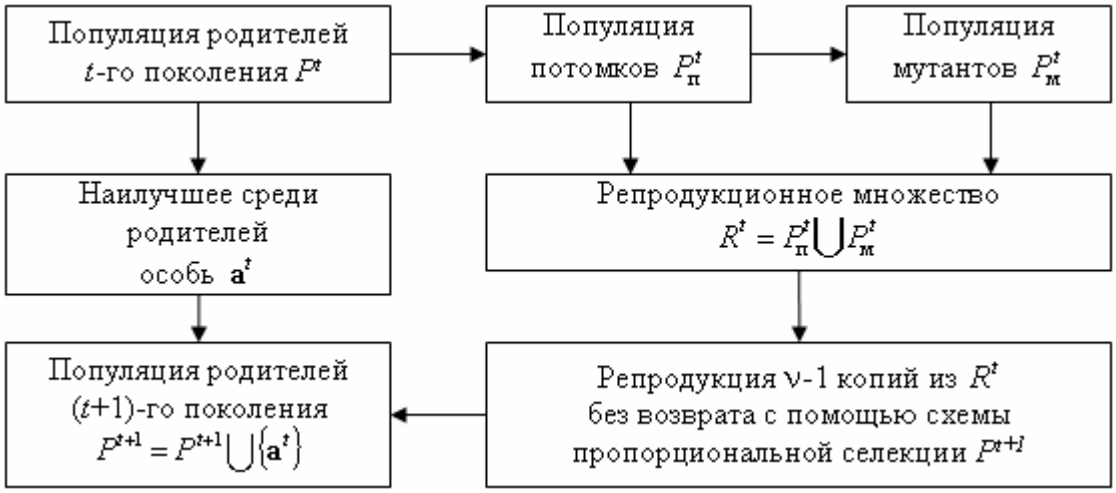

Рис. 2.2. Элитная поколенческая репродукция

Если интервал перекрытия поколений $G = \nu'/\nu \ \left(1 \le \nu' < \nu\right)$, то только n' новых особей формируются в популяции P^{t+1} путем копирования их из совокупности особей потомков и решений мутантов, воспроизведенных в *t*-м поколении, а все остальные $\left(\nu - \nu'\right)$ особей копируются в популяцию P^{t+1} из популяции P^t (рис. 2.3). Такая стратегия формирования популяции P^{t+1} из популяции P^t называется репродукцией *устойчивого состояния* (*strategic of steady state*). Согласно выражению (2.8), только n' особей в популяции P^{t+1} отличаются от особей родителей, составляющих популяцию P^t. Это позволяет говорить, что популяции P^t и P^{t+1} являются *частично перекрывающимися популяциями* (*overlapping populations*).

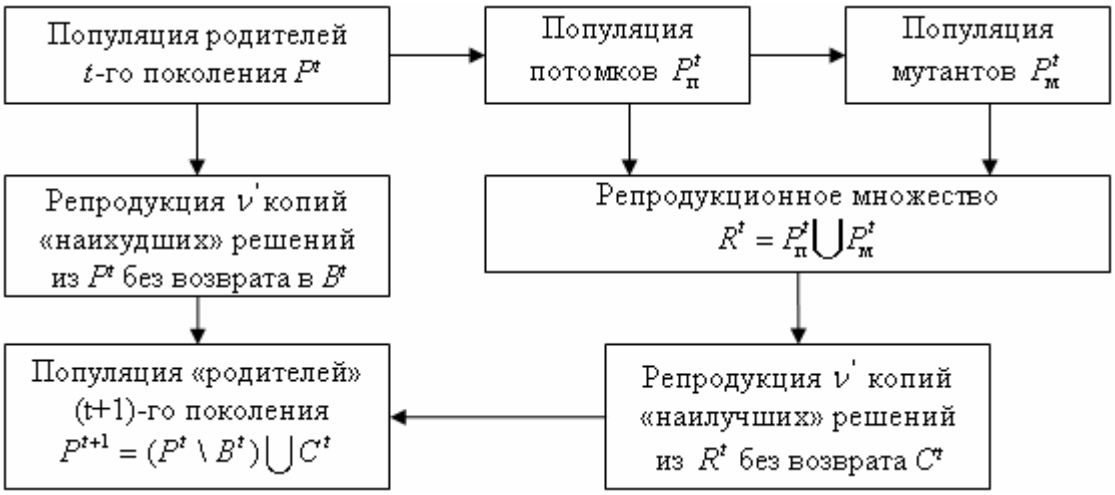

Рис. 2.3. Репродукции устойчивого состояния

Если $G = 1/\nu$, то популяция P^{t+1} отличается от популяции P^t только одной особью. В связи с этим, для того чтобы «наилучшую» особь из репродукционного множества R^t ввести в популяцию P^{t+1}, в последней следует освободить для нее место. Это может быть выполнено при помощи одной из следующих процедур:

- случайного исключения без возвращения с равной вероятностью $1/\nu$ любого решения из P^t;
- случайного исключения из популяции P^t без возвращения решения согласно обратной приспособленности

$$p(a_i^t) = \frac{\left(1/m(a_i^t)\right)}{\sum_{l=1}^{n}\left(1/m(a_i^t)\right)}, i = \overline{1,n}; \qquad (2.9)$$

- детерминированного исключения «наихудшей» особи P^t.

Например, из популяции P^t детерминированным образом исключается «наихудшая» особь, а из совокупности потомков и мутантов выбирается случайным образом согласно схеме пропорциональной селекции новое «наилучшая» особь и помещается на освободившееся в

популяции P^t место. При этом в схеме репродукции устойчивого состояния (см. рис. 2.3) имеет место равенство $\left|B^t\right| = \left|C^t\right| = 1$.

Процедуры выбора «наихудшего» решения из популяции P^t и «наилучшего» решения из репродукционного множества R^t, учитывающие не только значения функции приспособленности $m(a_i^t)$, но и структуру хромосом c_i^t, можно представить с помощью решения такой экстремальной задачи:

$$d(a^t, a_i^t) = \min_{l=1,n} d(c(a^t), c(a_l^t)) \qquad (2.10)$$

при условии, что

$m(a_l^t) < m(a^t)$,

где a^t – «лучшая» особь в популяции P^t,

a_i^t – особь, исключаемая из популяции P^t,

$d(c(a^t), c(a_l^t))$ – мера «близости» генотипов особей.

Из экстремальной задачи (2.10) видно, что особь, исключаемая из популяции, должна иметь невысокое значение приспособленности и быть как можно ближе к кодировке «лучшей особи» с точки зрения значения совпадения соответствующих значений генов.

В заключение рассмотрим общую стратегию формирования популяции P^{t+1}, описываемую с помощью выражения:

$$P^{t+1} = \left(P^t \setminus B^t\right) \cup C^t \qquad (2.11)$$

для любых значений параметра $G = v'/v \ \left(1 \le v' < v\right)$ из популяции P^t, в которой некоторые особи могут иметь одинаковые значения приспособленности (будем называть их *решениями i-го типа*).

Пусть $m_{i,t}$ – число решений i-го типа, которые имеют место в популяции P^t t-го поколения $(m_{i,t} \ge 1)$.

Предположим, что популяция B^t численностью $g = G \cdot v = v'$ формируется с помощью случайного выбора решений из популяции P^t с

равной вероятностью $1/v$, а популяция C^t той же численностью $g = G \cdot v = v'$ формируется из репродукционного множества R^t по схеме пропорциональной селекции согласно распределению вероятностей

$$\left\{ \frac{m(a_i)}{\sum_{a \in R^t} m(a)},\ a_i \in R^t \right\}. \qquad (2.12)$$

Для простоты будем считать, что $\left|R^t\right| = v$.

Фундаментальное уравнение популяционной генетики, связывающее ожидаемое число решений i-го типа в популяции P^{t+1} $(t+1)$-го поколения $\left(m_{i,t+1}\right)$ с ожидаемым числом решений i-го типа, копируемых из популяции P^t в популяцию B^t $\left(m_{i,t}^B\right)$, и ожидаемым числом решений i-го типа, копируемых из репродукционного множества R^t в популяцию C^t $\left(m_{i,t}^C\right)$ согласно выражению (2.11) можно записать следующим образом:

$$m_{i,t+1} = m_{i,t} + m_{i,t}^C - m_{i,t}^B. \qquad (2.13)$$

Это уравнение получило название *уравнения жизни, рождения и гибели*.

Учитывая предположения, сделанные выше, уравнение (2.13) можно записать в следующем виде:

$$m_{i,t+1} = m_{i,t} + m_{i,t} \cdot g \cdot \frac{m_i}{\sum_{a \in R^t} m(a)} - m_{i,t} \cdot g \cdot \frac{1}{n}. \qquad (2.14)$$

После некоторых преобразований получаем из (2.8), что

$$m_{i,t+1} = m_{i,t} \cdot \left(1 + G\left(\frac{m_i}{m_{cp}(t)} - 1\right)\right), \qquad (2.15)$$

где $m_{cp}(t) = \frac{1}{|R^t|}\sum_{a \in R^t} m(a)$ – среднее значение функции приспособленности для репродукционного множества R^t. Полученное уравнение (2.15) называется *уравнением роста решений i-го типа* при формировании популяции P^{t+1}.

При G=1 (поколенческая репродукция) уравнение (2.15) приобретает следующий вид:

$$m_{i,t+1} = m_{i,t} \cdot \frac{m_i}{m_{cp}(t)}. \tag{2.16}$$

Из уравнения (2.16) следует, что ожидаемое число решений i-го типа в популяции P^{t+1} при $m_i > m_{cp}(t)$ увеличивается, а при $m_i < m_{cp}(t)$ убывает, что справедливо и для $m_{i,t} = 1$.

Рассмотрим t последовательных поколений, начиная с t-го поколения, для которого справедливо выражение (2.16). Для следующего поколения имеет место выражение

$$m_{i,t+2} = m_{i,t+1} \cdot \left(1 + G\left(\frac{m_i}{m_{cp}(t+1)} - 1\right)\right). \tag{2.17}$$

Подставляя в (2.17) значение $m_{i,t+1}$ из (2.15), получаем

$$m_{i,t+2} = m_{i,t} \cdot \left(1 + G\left(\frac{m_i}{m_{cp}(t)} - 1\right)\right) \cdot \left(1 + G\left(\frac{m_i}{m_{cp}(t+1)} - 1\right)\right).$$

Поступая аналогичным образом, после t поколений можем записать, что ожидаемое число решений i-го типа в $(t+t)$-м поколении определяется с помощью следующего выражения:

$$m_{i,t+t} = m_{i,t} \cdot \prod_{k=0}^{t-1}\left(1 + G\left(\frac{m_i}{m_{cp}(t+k)} - 1\right)\right). \tag{2.18}$$

Предположим, что в течение t поколений отношение $\frac{m_i}{m_{cp}(t)} = const$ для всех $k = \overline{0,t}$, тогда из уравнения (2.18) получаем:

$$m_{i,t+t} = m_{i,t} \cdot \left(1 + G\left(\frac{m_i}{m_{cp}(t)} - 1\right)\right)^t . \qquad (2.19)$$

При $G = 1$ (поколенческая репродукция) из выражения (2.19) следует, что

$$m_{i,t+1} = m_{i,t} \cdot \left(\frac{m_i}{m_{cp}(t)}\right)^t . \qquad (2.20)$$

Следовательно, после t поколений ожидаемое число решений i-го типа при $m_i > m_{cp}(t)$ экспоненциально растет, а при $m_i < m_{cp}(t)$ экспоненциально убывает.

2.3.2. Схемы селекции

Селекция является одним из важнейших компонентов генетического алгоритма, который отличает последний от других алгоритмов поиска. Целью селекции является вероятностный отбор из репродукционного множества R^t (или популяции P^t) тех особей a_i^t, которые имеют высокие значения функции приспособленности $m(a_i^t)$, т.е. в рамках решения задачи поиска наиболее приспособлены к внешней среде, заданной с помощью критерия оптимальности Q. Будем называть такие особи «*наилучшими*» особями, а кодировки, соответствующие им – «*наилучшими*» *кодировками*. Аналогично, особи имеющие низкие значения функции приспособленности, будем называть *наименее приспособленными* или «*наихудшими*» *особями*, а соответствующие им кодировки – «*наихудшими*» *кодировками*.

По сути дела, оператор селекции реализует основой принцип *Дарвиновского естественного отбора*, заключающийся в том, что «выживают наиболее приспособленные, а наименее приспособленные

погибают». При этом осуществляется направленная селекция, приводящая к тому, что «наилучшие» кодировки копируются в популяцию P^{t+1} следующего поколения чаще, чем «наихудшие». Здесь под *гибелью кодировок* понимается удаление «наихудших» особей из популяции P^{t+1} и, следовательно, отстранение их от участия в образовании брачных пар и от воспроизводства «потомков».

На генетическом уровне оператор селекции можно рассматривать как процедуру, посредством которой проводится дифференциация генетической информации и наиболее приспособленные кодировки передаются из поколения в поколение, в то время как гены наименее приспособленных кодировок погибают.

Будем говорить, что процедура селекции обладает *инвариантой масштабирования* (*scale invariant*), если умножение функции приспособленности $m(a_i^t)$ на положительную константу K не изменяет вероятности репродукции.

Будем говорить, что процедура селекции обладает *инвариантой сдвига* (*translation invariant*), если суммирование функции приспособленности $m(a_i^t)$ с положительной константой K не изменяет вероятности репродукции.

Оператор селекции включает в себя два этапа.

Этап 1. Здесь определяется *ожидаемое число копий каждой i-ой кодировки* $c_i^t \in R^t$ в популяции P^{t+1} следующего поколения: n_i^t, $i = \overline{1,v}$, где $v = \left|R^t\right|$.

Этап 2. Здесь генерируются новые решения популяции P^{t+1} путем указания конкретных кодировок $c_i^t \in R^t$, копируемых в P^{t+1} на основании соответствующего им ожидаемого числа копий n_i^t.

На первом этапе используется *алгоритм селекции*, а на втором – *алгоритм вероятностного отбора*.

Наиболее широко применяемым алгоритмом селекции является *схема пропорциональной селекции* (*fitness-proportionate selection*), получаемая из выражения (2.21) при условии, что $m_{i,t} = 1$:

$$n_i^t = \frac{m_i}{m_{cp}(t)}, \tag{2.21}$$

где $m_i = m(a_i^t)$ – значение функции приспособленности для i-ой особи; $m_{cp}(t) = \frac{1}{|R^t|}\sum_{a \in R^t} m(a)$ – среднее значение функции приспособленности для репродукционного множества R^t.

Теорема 2.1. Схема пропорциональной селекции обладает инвариантой масштабирования.

Доказательство.

$p_{селекции}(Km_i) = \frac{Km_i}{\sum_{l=1}^{v} Km_l} = \frac{m_i}{\sum_{l=1}^{v} m_l} = p_{селекции}(m_i)$. Теорема доказана.

Теорема 2.2. Схема пропорциональной селекции не обладает инвариантой сдвига.

Доказательство. $p_{селекции}(m_i) = p_{селекции}(m_i + K) \leftrightarrow \frac{m_i}{\sum_{l=1}^{v} m_l} = \frac{m_i + K}{\sum_{l=1}^{v}(m_l + K)} \leftrightarrow$

$$\leftrightarrow \frac{\sum_{l=1}^{v}(m_l + K)}{\sum_{l=1}^{v} m_l} = \frac{m_i + K}{m_i} \leftrightarrow \leftrightarrow 1 + \frac{\sum_{l=1}^{v} K}{\sum_{l=1}^{v} m_l} = 1 + \frac{K}{m_i} \leftrightarrow \frac{v}{\sum_{l=1}^{v} m_l} = \frac{1}{m_i} \leftrightarrow m_{cp} = m_i.$$

В последнем равенстве получили противоречие. Теорема доказана.

Недостатком схемы пропорциональной селекции является то, что наличие в репродукционном множестве R^t так называемых *суперкодировок* – кодировок со значениями функции приспособленности

много больше, чем среднее значение функции приспособленности по множеству R^t – приводит к *преждевременной сходимости* генетического алгоритма к локальному оптиуму. Суперкодировки мешают другим кодировкам быть выбранными для копирования в популяцию P^{t+1} следующего поколения с целью производить в $(t+1)$-ом поколении какие-либо копии, что приводит к гибели многих потенциально возможных решений.

Одним из способов предотвращения преждевременной сходимости является использование *схемы линейной ранговой селекции* (*scheme of linear rank selection*), основанной на том, что в ней вместо значений функции приспособленности $m_i = m(a_i^t)$ используются ранги r_i^t, $i = \overline{1,v}$:

$$n_i^t = n^- + \left(n^+ - n^-\right) \cdot \frac{r_i^t - 1}{n - 1}, \qquad (2.22)$$

где $r_i^t = i$ – *ранг i-й кодировки*, когда все особи множества R^t отсортированы в порядке монотонного неубывания значений функции приспособленности;

n^- – *нижняя граница ожидаемого числа копий* для «наихудшей» особи с наименьшим значением функции приспособленности, имеющей ранг $r_1^t = 1$;

n^+ – *верхняя граница ожидаемого числа копий* для «наилучшей» кодировки с наибольшим значением функции приспособленности, имеющей ранг $r_v^t = v$.

Для того чтобы определить численные значения граничных оценок ожидаемого числа копий n^- и n^+, потребуем, чтобы значения n_i^t, определяемые формулой (2.22), удовлетворяли следующим условиям:

1) значения n_i^t должны монотонно увеличиваться $\left(0 \le n_1^t < n_2^t < \ldots < n_v^t\right)$ по отношению к возрастанию значений функции приспособленности $\left(m_1 \le m_2 \le \ldots \le m_v\right)$;

2) общая сумма ожидаемого числа копий решений, репродуцируемых в популяцию P^{t+1}, должна равняться численности n этой популяции

$$\sum_{i=1}^{v} n_i^t = v\,. \tag{2.23}$$

Подставим значение n_i^t из (2.17) в соотношение (2.18); учитывая, что $r_i^t = i$, получим:

$$n \cdot n^- + \frac{n^+ - n^-}{n-1} \cdot \sum_{i=1}^{n} (i-1) = n\,. \tag{2.24}$$

Так как $\sum_{i=1}^{v} (i-1) = v(v-1)/2$, то, подставляя это значение в (2.24), после некоторых преобразований получаем следующее соотношение:

$$n^- = 2 - n^+, \tag{2.25}$$

при выполнении которого второе требование (2.23) также будет выполняться.

В силу того, что ожидаемое число копий n_1^t согласно первому требованию должно принимать только неотрицательные значения: $n_1^t = n^- \geq 0$, из соотношения (2.25) получаем, что это условие выполняется, если имеет место неравенство:

$$n^+ \leq 2\,. \tag{2.26}$$

С другой стороны, при $r_v^t = n^+ > 1$ «наилучшая» особь (с максимальным значением функции приспособленности) должно иметь значение ожидаемого числа копий больше единицы:

$$n_n^t = n^+ > 1, \tag{2.27}$$

чтобы оно не было «потеряно» при формировании популяции P^{t+1}.

Таким образом, n^+ выбирается из интервала $(1,2]$ случайным образом или детерминированно, а n^- вычисляется по формуле (2.25).

Ожидаемое число копий n_i^t, репродуцируемых из R^t в P^{t+1}, может также оцениваться с помощью *схемы нелинейной ранговой селекции*:

$$n_i^+ = \frac{(1/a)^{r_i^t}}{\frac{1}{\nu}\sum_{k=1}^{\nu}(1/a)^{r_k^t}}, \quad i = \overline{1,\nu}, \tag{2.28}$$

где $a = const$ $(0 < a < 1)$.

Теорема 2.3. Линейная ранговая селекция обладает как инвариантой масштабирования, так и инвариантой сдвига.

Эти свойства ранговой селекции связаны с тем, что соответствующая позиция особи в отсортированном списке необработанных приспособленностей не затрагивается ни сдвигом, ни масштабированием обработанной приспособленности.

Схема *b -турнирной селекции* (*b -tournament selection*) организована несколько иначе, чем описанные выше схемы. Случайным образом с равной вероятностью из R^t выбирается группа из b особей. Среди выбранных в группу особей определяется «наилучшая» особь, копия которой репродуцируется в популяцию следующего поколения. Группа особей возвращается в R^t. Процедура выбора копии «наилучших» особей повторяется требуемое число раз. Число решений b называется *размером турнира*. Наиболее простым является *бинарный турнир*, проводимый между двумя решениями $(b = 2)$.

Теорема 2.4. b -турнирная селекция обладает как инвариантой масштабирования, так и инвариантой сдвига.

Доказательство. $p_{селекции}(a_k^t) = \frac{1}{\nu}$ и не зависит от значений функции приспособленности. Теорема доказана.

Вероятность отбора особи в процедуре Больцмановской селекции (*Boltzmann selection procedure*) рассчитывается по формуле (2.29). Особенность данной схемы селекции в том, что, обладая инвариантой сдвига, она не обладает инвариантой масштабирования.

$$p_{селекции}(a_i^t) = \frac{m_Б(a_i^t)}{\sum_{l=1}^{n} m_Б(a_l^t)},\ i = \overline{1,n}, \tag{2.29}$$

где $m_Б(a_i^t) = e^{\frac{m(a_i^t)}{T}}$. Здесь $T = const$ – параметр схемы селекции.

Теорема 2.5. Больцмановская селекция обладает инвариантой сдвига, но не обладает инвариантой масштабирования.

Доказательство.

$$p_{селекции}(m_i + K) = \frac{e^{(m_i+K)/T}}{\sum_{l=1}^{n} e^{(m_l+K)/T}} = \frac{e^{m_i/T} \cdot e^{K/T}}{\sum_{l=1}^{n} e^{m_l/T} \cdot e^{K/T}} = \frac{e^{m_i/T}}{\sum_{l=1}^{n} e^{m_l/T}} = p_{селекции}(m_i)$$

Таким образом, инварианта сдвига присутствует.

$$p_{селекции}(m_i) = p_{селекции}(Km_i) \leftrightarrow \frac{e^{m_i/T}}{\sum_{l=1}^{n} e^{m_l/T}} = \frac{e^{Km_i/T}}{\sum_{l=1}^{n} e^{Km_l/T}} \leftrightarrow K = 1.$$

Таким образом, в общем случае инварианта масштабирования отсутствует. Теорема доказана.

В схеме *селекции по степенному закону* (*power law selection procedure*) распределение вероятностей выбора особей в популяции имеет следующий вид:

$$p_{селекции}(a_i^t) = \frac{m_C(a_i^t)}{\sum_{l=1}^{n} m_C(a_l^t)},\ i = \overline{1,n}, \tag{2.30}$$

где $m_C(a_i^t) = \left(m(a_i^t)\right)^T$. Здесь $T = const$ – параметр схемы селекции.

Теорема 2.6. Селекция по степенному закону обладает инвариантой масштабирования, но не обладает инвариантой сдвига.

Доказательство.

$$p_{селекции}(m_i) = p_{селекции}(m_i + K) \leftrightarrow \frac{{m_i}^T}{\sum_{l=1}^{n} {m_l}^T} = \frac{(m_i + K)^T}{\sum_{l=1}^{n} (m_l + K)^T} \leftrightarrow$$

$\leftrightarrow K = 0$ (при $T > 0$).

Таким образом, в общем случае инварианта сдвига отсутствует.

$$p_{селекции}(Km_i) = \frac{(Km_i)^T}{\sum_{l=1}^{n} (Km_l)^T} = \frac{K^T {m_i}^T}{K^T \sum_{l=1}^{n} {m_l}^T} = \frac{{m_i}^T}{\sum_{l=1}^{n} {m_l}^T} = p_{селекции}(m_i).$$

Таким образом, инварианта масштабирования присутствует. Теорема доказана.

В процессе генетического поиска может проявляться ряд ситуаций, при которых представленные выше схемы селекции могут приводить к значительному снижению эффективности генетического поиска. Например, наличие в репродукционном множестве R^t суперкодировок приводит к *преждевременной сходимости*. Это объясняется тем, что ожидаемое число копий высокоприспособленных особей будет захватывать значительную часть популяции следующего поколения и приводить потере *генетического разнообразия хромосомного набора популяции*. Нередко возникает обратная ситуация, при когда значение средней приспособленности близко к значениям функции приспособленности совокупности особей. Такую ситуацию принято называть *интенсивностью конкуренции*, она приводит к тому, что оператору селекции трудно предпочесть одну кодировку другой и выделить среди них наиболее приспособленную.

Для разрешения подобных проблем можно воспользоваться методом линейного масштабирования (об этот методе пойдет речь в следующем

параграфе), который преобразует необработанную функцию приспособленности m в масштабируемую функцию приспособленности m_M, используя для этих целей линейный функционал.

Фактически инварианты схем селекции позволяют среди множества схем селекции выделить те, для которых применение метода линейного масштабирования будет оправдано. Так, если b-турнирная обладает инвариантой сдвига и инвариантой масштабирования, то никакой функционал линейного масштабирования не позволит исправить проблемную ситуацию, сложившуюся в процессе генетического поиска. С другой стороны, если используемая схема селекции не обладает какой-либо инвариантой, то для нее можно подобрать функционал линейного масштабирования, способного сместить акценты при отборе особей в популяцию следующего поколения и постепенно выправить проблемную ситуацию в процессе генетического поиска.

В сводной таблице 2.1 можно видеть описанные выше схемы селекции и их инварианты.

Таблица 2.1

Схемы селекции и их инварианты

Схема селекции	Инварианта сдвига	Инварианта масштабирования
Пропорциональная	-	+
Линейная ранговая	+	+
b-турнирная	+	+
Больцмановская	+	-
По степенному закону	-	+

На втором этапе оператора селекции обычно используется алгоритм вероятностного отбора, реализующий *схему выбора с помощью рулеточного колеса*. Основная идея этой схемы заключается в том, что каждой особи $a_i^t \in R^t$ назначается сектор рулеточного колеса, величина которого пропорциональна либо значению функции приспособленности $2p\dfrac{m_i}{\sum_{l=1}^{|R^t|} m_l}$, либо рангу $2p\dfrac{n_i^t}{n}$ в зависимости от использования на первом

этапе схемы пропорциональной селекции или схемы линейной (или нелинейной) ранговой селекции соответственно. Тогда особь $a_i^t \in R^t$ репродуцирует свою точную копию в популяцию P^{t+1} в том случае, если случайно выбранное число $x \in [0,2p]$ попадает в сектор, который соответствует i-ой кодировке. Каждый раз выбирается единственная копия для новой популяции P^{t+1}. В связи с этим алгоритм отбора повторяется столько раз, сколько кодировок генерируется вновь. При $G=1$ (поколенческая стратегия формирования P^{t+1}) алгоритм отбора повторяется n раз, при $G=n'/n$ (стратегия устойчивого состояния формирования P^{t+1}) алгоритм отбора повторяется n' раз (в частом случае – один раз для $n'=1$).

Метод "вращения рулеточного колеса" (или стохастический выбор с возвращением) в сочетании со схемой пропорциональной селекции можно представить в виде следующей процедуры.

1. Вычислить суммарную приспособленность особей $a_i^t \in R^t$:

$$F = \sum_{i=1}^{|R^t|} m_i \ .$$

2. Вычислить вероятность выбора каждой особи $a_i^t \in R^t$ в качестве копии, репродуцируемой в популяцию P^{t+1}:

$$p(a_i^t) = \frac{m_i}{F}, \ i = \overline{1,|R^t|} \ .$$

3. Вычислить совокупную вероятность для каждой особи $a_i^t \in R^t$:

$$p_i = \sum_{l=1}^{i} p(a_l^t) \ .$$

4. Случайным образом по равновероятному закону выбирается действительное число $x \in [0,2p]$.

5. Если $x < p_1$, то в популяцию P^{t+1} копируется особь a_1^t, иначе для копирования в популяцию P^{t+1} выбирается особь a_j^t, для которой выполняется условие:

$$p_{j-1} < x \le p_j . \tag{2.31}$$

При использовании этого алгоритма для формирования популяции P^{t+1} выбор очередной копии производится каждый раз из всего множества решений, образующих репродукционное множество R^t. При этом одно и то же решение может быть репродуцировано более одного раза.

Алгоритм вероятностного отбора, который позволяет определить все n копий, репродуцируемых из R^t в популяцию P^{t+1} с помощью единственного «вращения колеса», реализуется в виде однофазного алгоритма, называемого **стохастическим универсальным выбором**.

1. Определяется $N := \sum_{i=1}^{v} n_i^t$.

2. Осуществляется отображение всех решений из R^t в смежные отрезки на оси действительных чисел $[0, N)$ так, чтобы сегмент i-го решения равнялся величие его ожидаемого числа копий n_i^t.

3. Случайным образом с равной вероятностью генерируется n чисел $(x_1, x_2, ..., x_v)$, каждое из которых принадлежит интервалу $[0, N)$.

4. Для каждого i-го решения из R^t выбирается столько копий, сколько случайных чисел из $(x_1, ..., x_v)$ попало в i-й сегмент на интервале $[0, N)$.

В качестве алгоритмов вероятностного отбора, основанных на информации о совокупности ожидаемого числа копий n_i^t, $i = \overline{1, v}$, отметим следующие алгоритмы.

Остаточный стохастический выбор с возвращением

1. Пусть $n_i^t = \lfloor n_i^t \rfloor \cdot \{n_i^t\}$, $i = \overline{1,v}$, где $\lfloor n_i^t \rfloor$ – целая часть числа; $\{n_i^t\}$ – дробная часть числа n_i^t, являются действительными числами, определенными на первом этапе оператора селекции.

Целочисленная фаза.

2. Детерминированно в популяцию P^{t+1} репродуцируется $\lfloor n_i^t \rfloor$ копий для каждого решения из репродукционного множества R^t.

3. Определяется число недостающих копий $\bar{v}$, необходимых для заполнения популяции P^{t+1} полностью n копиями:

$$\bar{v} = v - \sum_{i=1}^{v} \lfloor n_i^t \rfloor.$$

4. Если $\bar{v} = 0$, то алгоритм завершает свою работу.

Дробная фаза.

5. Определяется $N := \sum_{i=1}^{n} \{n_i^t\}$.

6. Осуществляется отображение всех решений из R^t в смежные отрезки на оси действительных чисел $[0, N)$ так, чтобы каждый сегмент i-го решения равнялся дробной части $\{n_i^t\}$ его ожидаемого числа копий n_i^t.

7. Случайным образом с равной вероятностью генерируется число $x \in [0, N)$.

8. В качестве копии выбирается то решение, в отрезок которого попало число x.

9. Процедура стохастического выбора с возвращением повторяется с шага 7 до тех пор, пока не будет получено $\bar{v}$ копий.

Остаточный стохастический выбор без возвращения

Этот алгоритм вероятностного отбора состоит, так же как и предыдущий алгоритм, из двух фаз: целочисленной и дробной.

1. Выполняются шаги 1-4, включая целочисленную фазу предыдущего алгоритма.

2. Выполняются шаги 5-8 дробной фазы предыдущего алгоритма.

3. Для выбранного в качестве копии *i*-го решения его ожидаемое число копий n_i^t принимается равным нулю (этим решения в течение дробной фазы предохраняются от многократного выбора).

4. Процедура стохастического выбора с возвращением повторяется с шага дробной фазы предыдущего алгоритма до тех пор, пока не будет получено $\bar{v}$ копий.

При использовании этого алгоритма для формирования популяции P^{t+1} выбранное однажды для копирования решение удаляется из репродукционного множества R^t. Таким образом, популяция P^{t+1} не содержит повторяющихся копий.

Детерминированный выбор

Этот алгоритм, так же как и два предыдущих, состоит из двух фаз: целочисленной и дробной.

Целочисленная фаза.

1. Детерминированно в популяцию P^{t+1} репродуцируется $[n_i^t]$ копий для каждого решения из репродукционного множества R^t.

2. Определяется число недостающих копий $\bar{v}$, необходимых для заполнения популяции P^{t+1} полностью n копиями.

3. Если $\bar{v} = 0$, то алгоритм завершает свою работу.

Дробная фаза.

4. Решения репродукционного множества R^t упорядочиваются в порядке уменьшения дробной части $\{n_i^t\}$ их ожидаемого числа копий n_i^t.

5. Детерминированно в популяцию P^{t+1} отбираются первые $\bar{v}$ копий упорядоченных на 4-м шаге решений. После этого алгоритм заканчивает свою работу.

Оператор селекции включает в себя по одному алгоритму из следующих двух групп: *группы алгоритмов селекции* (схемы пропорциональной селекции; схемы линейной ранговой селекции; схемы нелинейной ранговой селекции) и *группы алгоритмов вероятностного отбора* (стохастический выбор с возвращением; стохастический универсальный выбор; остаточный стохастический выбор с возвращением; остаточный стохастический выбор без возвращения; детерминированный выбор), что позволяет использовать при формировании популяции P^{t+1} комбинированные схемы операторов селекции, реализующих *принцип естественного отбора*.

2.3.3. Масштабирование

Как отмечалось ранее, при решении задачи поиска (1.2) схема пропорциональной селекции при наличии в репродукционном множестве R^t суперкодировок приводит к *преждевременной сходимости*. Это объясняется тем, что ожидаемое число копий суперкодировки при поколенческой стратегии формирования популяции будет на втором этапе оператора селекции захватывать значительную часть популяции P^{t+1} следующего поколения. При этом поиск начинает концентрироваться в окрестности суперкодировки в силу потери *генетического разнообразия хромосомного набора популяции* P^{t+1}, игнорируя обработку других кодировок репродукционной группы, находящихся в других частях пространства поиска S.

Однако, даже сохраняя значительное хромосомное генетическое разнообразие, схема пропорциональной селекции может привести к потере конкуренции между кодировками репродукционной группы в том случае, когда значение средней приспособленности $m_{cp}(t)$ близко к значениям

функции приспособленности совокупности особей $a_i \in R^t$ (где $m(a_i) \cong m_{cp}(t)$). Это объясняется тем, что кодировки, имеющие приблизительно равные значения функции приспособленности, будут иметь почти одно и то же ожидаемое число копий в следующем поколении. Такая ситуация, называемая *интенсивностью конкуренции*, приводит к тому, что оператору селекции трудно предпочесть одну кодировку другой и выделить среди них наиболее приспособленную.

Компромисс между преждевременной сходимостью и выравниванием интенсивности конкуренции можно обеспечить с помощью *методов масштабирования*, которые преобразуют необработанную функцию приспособленности m в масштабируемую функцию приспособленности m_M.

Наиболее часто в генетических алгоритмах используется *линейное динамическое масштабирование*:

$$m_M(a) = a_t m(a) + b_t, \tag{2.32}$$

где $a_t > 0$ (для задачи максимизации); b_t – коэффициент масштабирования.

Коэффициенты a_t и b_t выбираются в каждом t-м поколении по информации о значениях функции приспособленности m для обрабатываемых кодировок $s \in R^t$, исходя из следующих двух условий.

1. Среднее значение масштабированной функции приспособленности для множества R^t:

$$\overline{m}_M^t = \frac{1}{|R^t|} \sum_{a \in R^t} m_M(a) \tag{2.33}$$

должно равняться среднему значению необработанной функции приспособленности (2.3) для множества R^t

$$\overline{m}_M^t = m_{cp}(t). \tag{2.34}$$

Учитывая (2.32), получим:

$$a_t m_{cp}(t) + b_t = m_{cp}(t). \qquad (2.35)$$

Выполнение равенства (2.35) гарантирует, что каждая кодировка $s \in R^t$, имеющая значение функции приспособленности совпадающее со значением средней приспособленности по популяции, при использовании схемы пропорциональной селекции репродуцирует в среднем одну ожидаемую копию в популяцию P^{t+1}.

2. Наибольшее значение масштабируемой функции приспособленности m_M^+ должно находиться в следующем отношении со средним значением необработанной функции приспособленности:

$$\frac{m_M^+}{m_{cp}(t)} = c, \qquad (2.36)$$

где c = *const*, $c \in [1.2;2.0]$ – ожидаемое число копий для особи, которая имеет максимальное значение необработанной функции приспособленности в R^t.

Выполнение равенства (2.36) гарантирует, что «наилучшая» кодировка множества R^t репродуцирует при использовании схемы пропорциональной селекции в среднем c ожидаемых копий в популяцию P^{t+1}.

Таким образом, решая уравнения (2.34) – (2.36), получаем, что для масштабируемой функции приспособленности (2.32) в каждом t-м поколении коэффициенты a_t и b_t должны вычисляться согласно следующим выражениям:

$$a_t = \frac{(c-1)\cdot m_{cp}(t)}{m^+(t) - m_{cp}(t)}; \qquad (2.37)$$

$$b_t = \frac{m_{cp}(t) \cdot \left(m^+(t) - c \cdot m_{cp}(t)\right)}{m^+(t) - m_{cp}(t)}. \qquad (2.38)$$

Здесь $m^+(t)$ – значение функции приспособленности «наилучшей» особи в R^t.

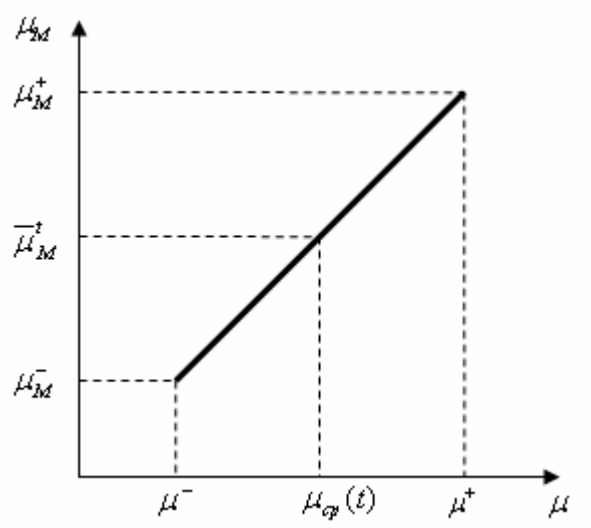

Рис. 2.4. Зависимость m_M от m для $c = 2$ и $m_M^- \geq 0$

Зависимость значений масштабируемой функции приспособленности m_M от значений необработанной функции приспособленности m приведена на рис. 2.4.

Одно из требований, предъявляемых к необработанной функции приспособленности m, состоит в том, что она должна принимать неотрицательные значения для всех $s \in S$. Это требование должно сохраняться и для масштабируемой функции приспособленности:

$$m_M(s) \geq 0 \ \forall s \in S. \qquad (2.39)$$

Однако для некоторых репродукционных множеств R^t масштабируемая функция приспособленности (2.36) с коэффициентами a_t и b_t из выражений (2.37) и (2.38) может принимать отрицательные значения (см. рис. 2.5). Такая ситуация возникает обычно, когда большинство кодировок репродукционного множества R^t являются

наиболее приспособленными, и их значения необработанной функции приспособленности $m(s)$ отличаются друг от друга очень мало, в то время как ряд кодировок этого же множества R^t имеет значения необработанной функции приспособленности значительно ниже среднего значения приспособленности $m_{cp}(t)$.

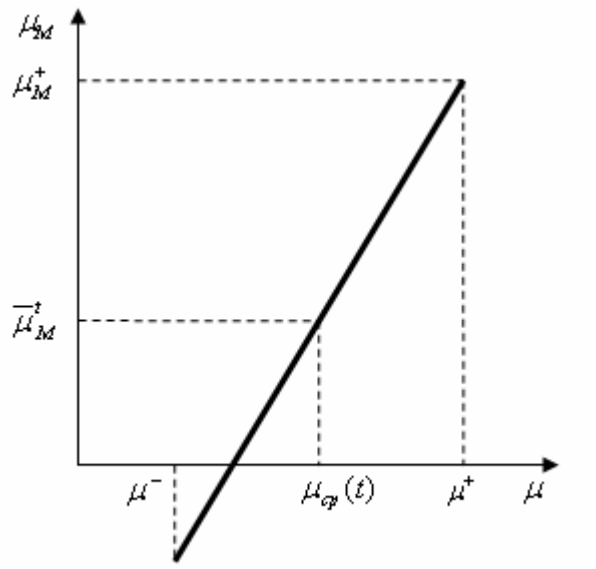

Рис. 2.5. Зависимость m_M от m для $c = 2$ и $m_M^- < 0$

Для решения проблемы исключения отрицательных значений масштабируемой функции приспособленности (2.32) потребуем, чтобы она удовлетворяла следующим двум условиям:

1) $\overline{m}_M^t = m_{cp}(t)$;

2) $m_M^- = 0$ – значение масштабированной функции приспособленности «наихудшей» кодировки.

С учетом этих условий находим, что коэффициенты a_t и b_t, обеспечивающие неотрицательные значения масштабируемой функции приспособленности, должны вычисляться согласно следующим выражениям:

$$a_t = \frac{m_{cp}(t)}{m_{cp}(t) - m^-(t)}; \tag{2.40}$$

$$b_t = \frac{m_{cp}(t) \cdot m^-(t)}{m_{cp}(t) - m^-(t)}. \tag{2.41}$$

Зависимость значений масштабированной функции приспособленности m_M от значений необработанной функции приспособленности приведена на рис. 2.6.

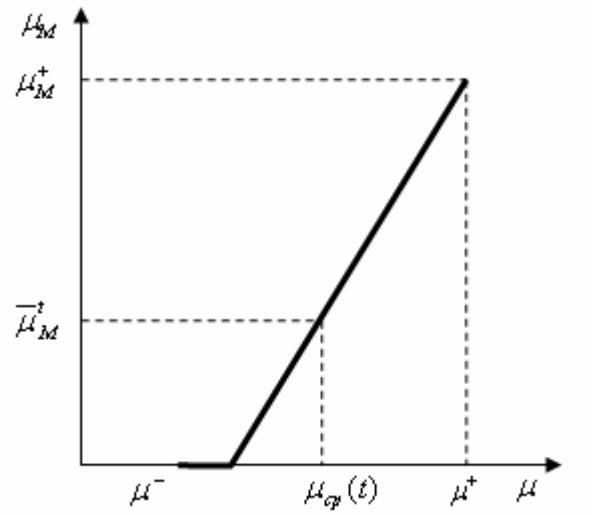

Рис. 2.6. Зависимость m_M от m для $c = 2$ и $m_M^- = 0$

Учитывая выражение (2.32), условие (2.39) для «наихудшей» кодировки из R^t можно записать в следующем виде:

$$a_t m^- + b_t \geq 0$$

или

$$m^- \geq \frac{b_t}{a_t}. \tag{2.42}$$

Тогда, подставляя в неравенство (2.42) значения коэффициентов a_t и b_t из выражений (2.37) (2.38), получаем неравенство:

$$m^- \geq \frac{c \cdot m_{cp}(t) - m^+}{c-1}. \qquad (2.43)$$

Таким образом, при линейном динамическом масштабировании (2.32), если неравенство (2.43) для заданного значения $c \in [1.2;\ 2.0]$ выполняется, то коэффициенты a_t и b_t вычисляются с помощью выражений (2.37), (2.38), в противном случае коэффициенты a_t и b_t вычисляются с помощью выражений (2.40), (2.41).

Кроме линейного динамического масштабирования, преобразование значений функции приспособленности, оценивающих решения из репродукционного множества R^t, может осуществляться с помощью других методов. Приведем некоторые из них.

Масштабирование по степенному закону

$$m_M(s) = (m(s))^a, \qquad (2.44)$$

где $a > 1$ – константа масштабирования (например, $a = 1.05$).

При $a = 2$ автоматически обеспечивается неотрицательность значений масштабированной функции приспособленности.

Масштабирование с помощью логарифмической функции

$$m_M(s) = C - \log m(s), \qquad (2.45)$$

где $C = const$, удовлетворяющая условию $C > \log m(s)$ для всех $s \in S$.

Сигма-усеченное масштабирование

$$m_M(s) = m(s) - \left(m_{cp}(t) - C \cdot s(t)\right), \tag{2.46}$$

где $C = const$, выбираемая в интервале [1,3];

$s(t) = \sqrt{\frac{1}{v-1}\sum_{i=1}^{v}\left(m_i - m_{cp}(t)\right)^2}$ – стандарт отклонения от среднего значения в множестве R^t.

Если $m_i + C \cdot s(t) < m_{cp}(t)$, то считается, что $m_M = 0$.

Больцмановское масштабирование

$$m_M(s) = e^{\left(\frac{m(s)}{T}\right)}, \tag{2.47}$$

где $T = const$, которая может изменяться в процессе генетического поиска.

Для каждого из приведенных методов масштабирования значений функции приспособленности справедлива схема пропорциональной селекции (2.21), определяющая число репродуцированных копий кодировок из репродукционного множества R^t в популяцию P^{t+1} следующего $(t+1)$-го поколения.

2.4. Классические генетические операторы кроссовера

Кроссовер — генетический оператор скрещивания, осуществляющий обмен генами или частями хромосом между двумя родителями с целью получения новых хромосом, являющихся кодировками их потомков.

При этом генетический материал из хромосом родителей согласно свойству наследственности непосредственно передается в хромосомы потомков.

Оператор кроссовера является настолько важным оператором ГА, что если его исключить из структуры алгоритма, то генетический поиск потеряет свойства, присущие репродукционно-популяционным алгоритмам: воспроизводить новые кодировки, наследующие генетический материал из кодировок-родителей.

Оператор кроссовера применяется к кодировкам «брачной пары» с вероятностью p_c. Этот параметр называется *скоростью кроссовера* и характеризует вероятность того, что кроссовер имеет место (обычно $p_c \in [0.5;1]$). Если кроссовер для кодировок «брачной пары» не состоялся, то генерируются два потомка, которые являются точными копиями своих родителей (можно потребовать, чтобы потомки в этом случае просто не репродуцировались или генерировался один из родителей с меньшим значением функции приспособленности). При заданном числе возможных «брачных пар» $N_{БП}$ *ожидаемое число актов кроссовера* равно $N_{БП} \cdot p_c$, что позволяет сгенерировать в течение каждого поколения в среднем $2 \cdot N_{БП} \cdot p_c$ потомков.

Под *размножением* понимается процесс формирования в t-м поколении по генетической информации каждой из родительских кодировок одной или нескольких новых кодировок потомков, которые обладают преемственностью наследственных свойств родителей.

Рассмотрим ряд классических алгоритмов кроссовера, используемых в процессе размножения кодировок, которые могут быть заданы в форме как бинарного, так и n-арного представления. Механизм размножения с помощью этих операторов кроссовера прост, т. к. содержит следующие *простейшие операции*:

- генерация случайных чисел;

- копирование строк фиксированной длины L, являющихся «родительскими» кодировками;
- обмен кусками «родительских» кодировок, разрываемых в одной и той же точке хромосомы.

Применение этих операторов приводит к тому, что кодировки-потомки будут содержать новые сочетания аллелей генов, принадлежащих кодировкам родителей, т.е. происходит только перераспределение существующих аллелей родителей по хромосомам потомков.

Наиболее широко применяемым оператором кроссовера является *одноточечный кроссовер*, основная идея которого состоит в следующем.

1. Согласно выбранной системе скрещивания формируется «брачная пара» из допустимых родительских кодировок $c' = (c'_1,...,c'_L)$ и $c'' = (c''_1,...,c''_L)$.
2. Случайным образом с равной вероятностью $\frac{1}{L-1}$ выбирается *точка разрыва* $r \in \{1,2,...,L-1\}$.
3. Кодировки c' и c'' разрываются в одной и той же точке r.
4. Из полученных четырех кусков родительских кодировок

$$(c'_1,...,c'_r),\ (c'_{r+1},...,c'_L),\ (c''_1,...,c''_r),\ (c''_{r+1},...,c''_L)$$

формируются кодировки двух потомков

$$c^1 = (c'_1,...,c'_r),\ (c''_{r+1},...,c''_L),$$

$$c^2 = (c''_1,...,c''_r),\ (c'_{r+1},...,c'_L).$$

Если гены в родительских хромосомах имеют одинаковые аллели ($c'_i = c''_i = x$) в одном и том же i-м локусе, то и воспроизведенные ими потомки будут иметь в i-м локусе ту же самую аллель ($c^1_i = c^2_i = x$). Используя этот факт, потребуем, чтобы точка разрыва r в одноточечном кроссовере выбиралась случайным образом с равной вероятностью из подынтервала $r^- \le r \le r^+$ ($[r^-, r^+] \subseteq [1, L-1]$), где

$$r^- = \min_{1 \le j \le (L-1)} \left\{ j \in J \mid c'_j \neq c''_j \right\},$$

$$r^+ = \max_{2 \le j \le (L-1)} \left\{ j \in J \mid c'_j \neq c''_j \right\}.$$

Здесь r^- – номер первого встречающегося локуса в хромосомах родителей c', c'', в котором аллели родительских генов не совпадают между собой; r^+ – номер последнего встречающегося локуса в хромосомах родителей c', c'', в котором аллели родительских генов не совпадают между собой.

Оператор кроссовера, реализующий эту стратегию размножения, называется *ограниченным одноточечным кроссовером*.

В том случае когда «родительские» кодировки c', c'' разрываются в двух точках r_1 и r_2 ($r_1 < r_2$), случайным образом выбранных с равной вероятностью без возвращения из интервала [1, (L–1)], оператор кроссовера называется *двухточечным кроссовером*.

Модификацией двухточечного кроссовера является *ограниченный двухточечный кроссовер*, в котором точки разрыва r_1 и r_2 ($r_1 < r_2$) выбираются случайным образом с равной вероятностью из интервала [r^-, r^+), определяемого таким же образом, как и в ограниченном одноточечном кроссовере.

Оператор кроссовера, в котором единственный потомок воспроизводится таким образом, что аллель каждого гена копируется в хромосому потомка либо из хромосомы одного родителя, либо из хромосомы другого родителя с вероятностью $\frac{1}{2}$, называется *однородным кроссовером*.

Модификацией однородного кроссовера является однородный кроссовер, учитывающий приспособленности родителей, в котором копирование аллелей из родительских генов в хромосому потомка c^1 осуществляется согласно следующему распределению вероятностей:

$$c_j^1 = \begin{cases} c'_j & \text{с вероятностью } P(c'_j) = \dfrac{m(c')}{m(c') + m(c'')}; \\ c''_j & \text{с вероятностью } P(c''_j) = 1 - P(c'), \end{cases}$$

где $m(c')$, $m(c'')$ – значения функции приспособленности, оценивающие, соответственно, родительские кодировки c' и c''.

Другой классический кроссовер, *многоточечный*, интерпретирует хромосому как кольцо, которое точки разрыва разрезают на сегменты. На рисунке 2.7 показана схема работы такого кроссовера для 4-х точечного случая.

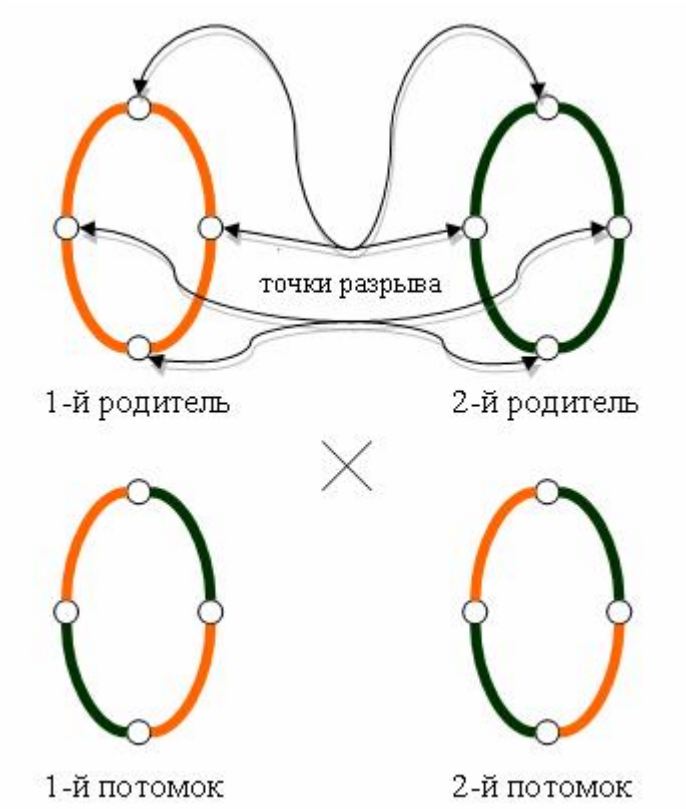

Рис. 2.7. Схема работы 4-х точечного кроссовера.
Многоточечный кроссовер трактует хромосому как кольцо

2.5. Классические генетические операторы мутации

Как отмечалось выше, генетическая информация (значения аллелей, содержащихся в хромосомном наборе популяции P^t) не меняется в кодировках потомков, которые построены с помощью операторов кроссовера. Источником генетической изменчивости кодировок-потомков являются мутации – изменения аллелей в одном или нескольких генах кодировок

потомков в результате появления в них новых аллелей, не содержащихся ранее в соответствующих генах. Кодировку, полученную в результате мутации, будем называть *кодировкой-мутантом*.

В том случае если кодировка c представлена в бинарном алфавите $\{0,1\}$ в виде n-битовых строк, наиболее простым видом мутации является *точечная мутация*, связанная с изменением значения аллели в случайно выбранном гене на обратное («1» на «0» или «0» на «1»). С геометрической точки зрения точечная мутация — это случайное блуждание, не зависящее от значения функции приспособленности, по 2^n вершинам n-мерного гиперкуба, что обеспечивает ненулевую вероятность достижения любой кодировки c в пространстве поиска S.

Оператор точечной мутации применяется к кодировкам-потомкам с вероятностью p_m. Этот параметр называется скоростью мутации и характеризует вероятность того, что мутация имеет место (обычно $p_m \in [0.001, 0.01]$). При заданном числе возможных мутаций N_m ожидаемое число актов мутации равно $N_m \cdot p_m$.

В качестве операторов мутации можно использовать также следующие две операции, выполняемые над n-битовой кодировкой $c = (c_1,...,c_L)$:

- *дополнение*: кодировка мутанта получается путем изменения значений аллелей во всех генах кодировки c на обратные («1» на «0» или «0» на «1»);
- *инверсия*: кодировка c разрывается в двух точках r_1 и r_2 ($r_1 < r_2$), случайно выбранных с равной вероятностью из интервала $[1, (L-1)]$; значения аллелей во всех генах куска изменяются на обратные («1» на «0» или «0» на «1»); кодировка мутанта c^M образуется путем соединения вновь двух старых кусков и нового куска

$$(\overline{c}_{r_1+1},...,\overline{c}_{r_2}):\ c^M = (c_1,...,c_{r_1},\overline{c}_{r_1+1},...,\overline{c}_{r_2},c_{r_2+1},...,c_L);$$

- *сальтация*: в кодировке c случайным образом инвертируются одновременно значения аллелей в k $(k < L)$ генах родительской хромосомы;

- *транслокация*: в кодировке c выделяется не один кусок $[r_1, r_2]$, а сразу несколько непересекающихся между собой участков «родительской» хромосомы c ; значения аллелей в мутанте внутри этих участков инвертируются.

2.5.Операторы кроссовера и мутации для порядкового представления

Как уже было сказано выше, порядковое представление является декодером, и для обеспечения допустимости получаемых потомков требует применения специальных операторов. Приведем некоторые из возможных операторов для перестановочного представления.

2.5.1.Кроссовер порядка (кроссовер OX)

Кроссовер порядка (*кроссовер OX*), используя генотипы двух родителей в виде перестановок s_P^1 и s_P^2, создает генотипы пары потомков в виде новых допустимых перестановок $s_П^1$ и $s_П^2$.

Генотипы потомков сохраняют *порядок*[1] и *позицию*[2] имен городов в последовательности одного из родителей, в то же время они предохраняют от нарушения относительный порядок оставшихся имен городов от другого родителя.

Основная идея кроссовера порядка заключается в том, что важны порядок городов, а не их позиции.

Алгоритм кроссовера OX

1. Случайным образом с вероятностью $\frac{1}{n}$ из популяции P^t, содержащей n Гамильтоновых циклов, выбираются две «родительские» особи, генотипы которых являются перестановками s_P^1 и s_P^2:

   s_P^1 – перестановка, определяющая генотип 1-го родителя, которая называется разрезаемой строкой (cutting string);

   s_P^2 – перестановка, определяющая генотип 2-го родителя, которая называется заполняемой строкой (filler string).

2. Случайным образом с вероятностью $\frac{1}{n-1}$, где n – общее число городов, выбираются две точки a_1 и a_2 $(a_1 < a_2;\ a_1, a_2 \in [1, n-1])$, называемые *точками кроссовера* (*crossover points*), которые разрезают генотипы родителей на три части.

---

[1] Порядок – номера городов, следующие один за другим в конкретной перестановке.

[2] Позиция – номер локуса конкретной перестановки, где стоит номер города.

3. Непрерывная подпоследовательность имен городов из разрезаемой сроки, называемая *секцией кроссовера* (*crossover section*), которая находится между точками кроссовера a_1 и a_2, полностью копируется в генотип потомка $s_П^1$ в те же самые абсолютные позиции, которые имели место в разрезаемой строке, т.е. порядок и позиции городов из секции кроссовера сохраняются в $s_П^1$.

4. Формируется *блок заполнения* (*filler block*) из имен городов заполняемой строки s_P^2, которых нет в секции кроссовера разрезаемой строки уже скопированных в $s_П^1$ из s_P^1.

5. Элементы блока заполнения наследуются из заполняемой строки, основываясь на относительном а не на абсолютном порядке в ней, начиная с начала заполняемой строки s_P^2 (с позиции 1). Генотип потомка в виде перестановки $s_П^1$ конструируется при помощи добавления элементов из блока заполнения к секции кроссовера разрезаемой строки, находящейся уже в $s_П^1$:

   5.1. первый элемент блока заполнения помещается в позицию следующую сразу после точки кроссовера a_2;

   5.2. последующие за ним элементы блока заполнения занимают свободные позиции в относительном порядке;

   5.3. когда конец строки $s_П^1$ будет достигнут, процесс заполнения продолжается с первой свободной позиции слева от конца строки $s_П^1$ и двигаясь к точке кроссовера a_1 до тех пор пока все позиции строки $s_П^1$ не будут заполнены полностью.

6. Генотип второго потомка $s_П^2$ формируется аналогичным образом для родителей s_P^1 и s_P^2, роли которых поменялись:

   s_P^1 – перестановка, определяющая генотип 1-го родителя, стала заполняемой строкой;

   s_P^2 – перестановка, определяющая генотип 2-го родителя, стала разрезаемой строкой.

   Ниже показан пример (см. рис. 2.8.) работы алгоритма.

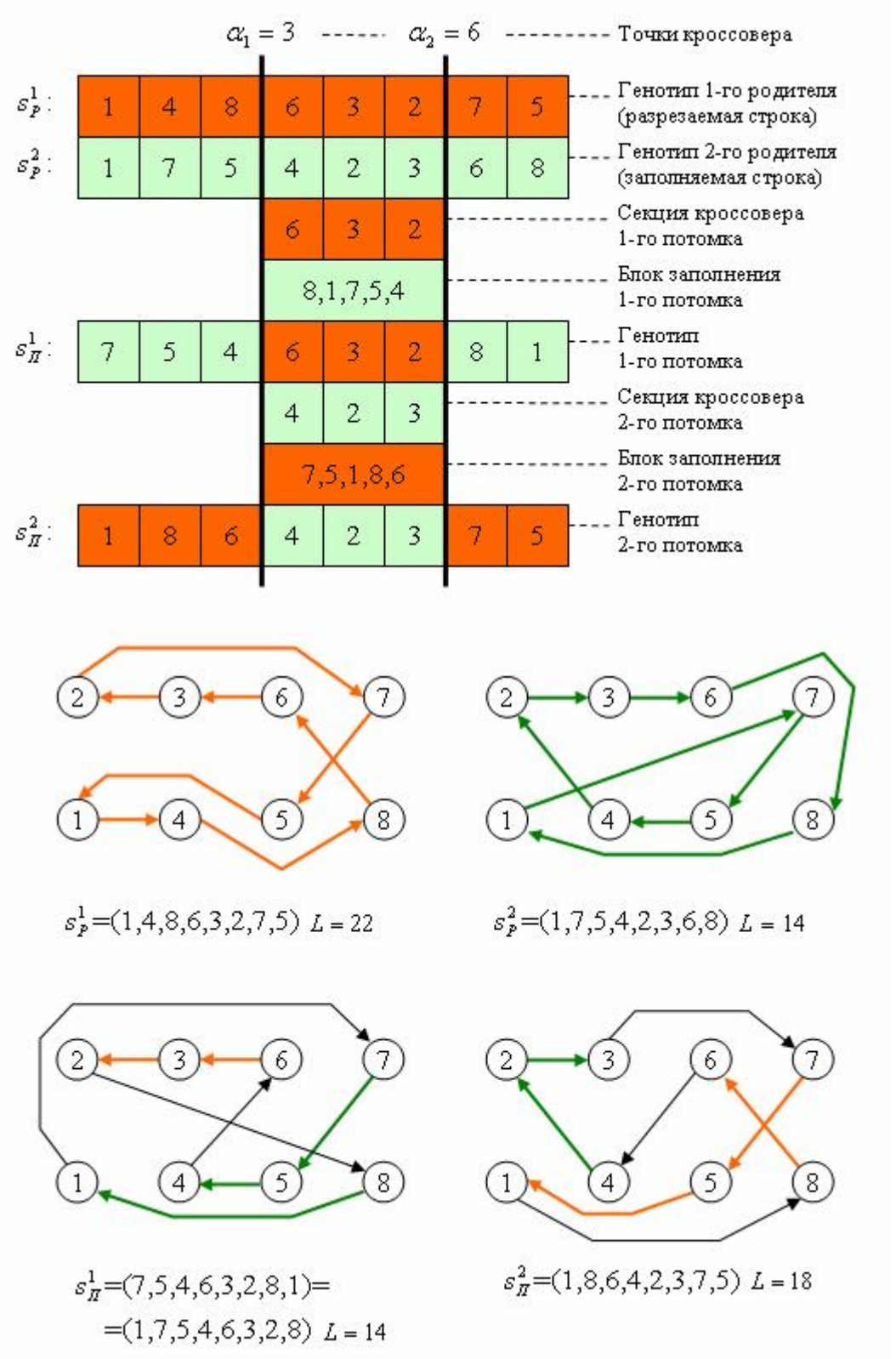

Рис. 2.8. Из двух родителей кроссовером OX получаем двух потомков (см. в приложении задачу 1). Ниже представлены их Гамильтоновы циклы.

2.5.2. Циклический кроссовер

Циклический кроссовер (*cycle crossover*) или *кроссовер CX* формирует генотип потомка $s_{П}$ таким образом, чтобы каждый город и его позиция переходили от генотипа одного из родителей. При этом он сохраняет в потомке абсолютные позиции городов из «родительских» генотипов, заданных перестановками s_P^1 и s_P^2. Поэтому кроссовер CX по своей идеи относится к классу позиционных кроссоверов.

Для обеспечения с помощью циклического кроссовера допустимого генотипа $s_{П}$ должны удовлетворяться следующие условия.

1. Каждая позиция генотипа потомка обязана сохранить имя города, найденного в соответствующей позиции одного из родителей.
2. Генотип потомка является перестановкой без пропусков и дублирования города.

Алгоритм циклического кроссовера

1. Выбираются из популяции P^t любые две «родительские» особи, генотипы которых заданы перестановками s_P^1 и s_P^2. Одна из них называется *базовой особью* (пусть ей соответствует перестановка s_P^1).
2. Формируется *вектор меток циклов* $\vec{c} = (c_1, c_1, ..., c_n)$, все компоненты которого равны нулю $c_i = 0, i = \overline{1,n}$.
3. Будем обозначать через k номер текущего строящегося цикла. Пусть k=0.
4. k=k+1.
5. Случайным образом из генотипа базовой особи s_P^1 из не посещенных городов выбирается j-ый город с именем $s_P^1(j)$. Данный город будет служить началом конструируемого цикла (первый город цикла).
6. Отмечаем в векторе $\vec{c}$ тот факт, что j-ый город в s_P^1 попал в k-ый цикл: $c_j = k$. Теперь j-ый город в s_P^1 считаем посещенным.
7. Рассчитываем индекс j^* как позицию города с именем $i = s_P^2(j)$ в строке s_P^1, т.е. j^* такое, что $s_p^1(j^*) = s_p^2(j)$.
8. Принимаем $j = j^*$.

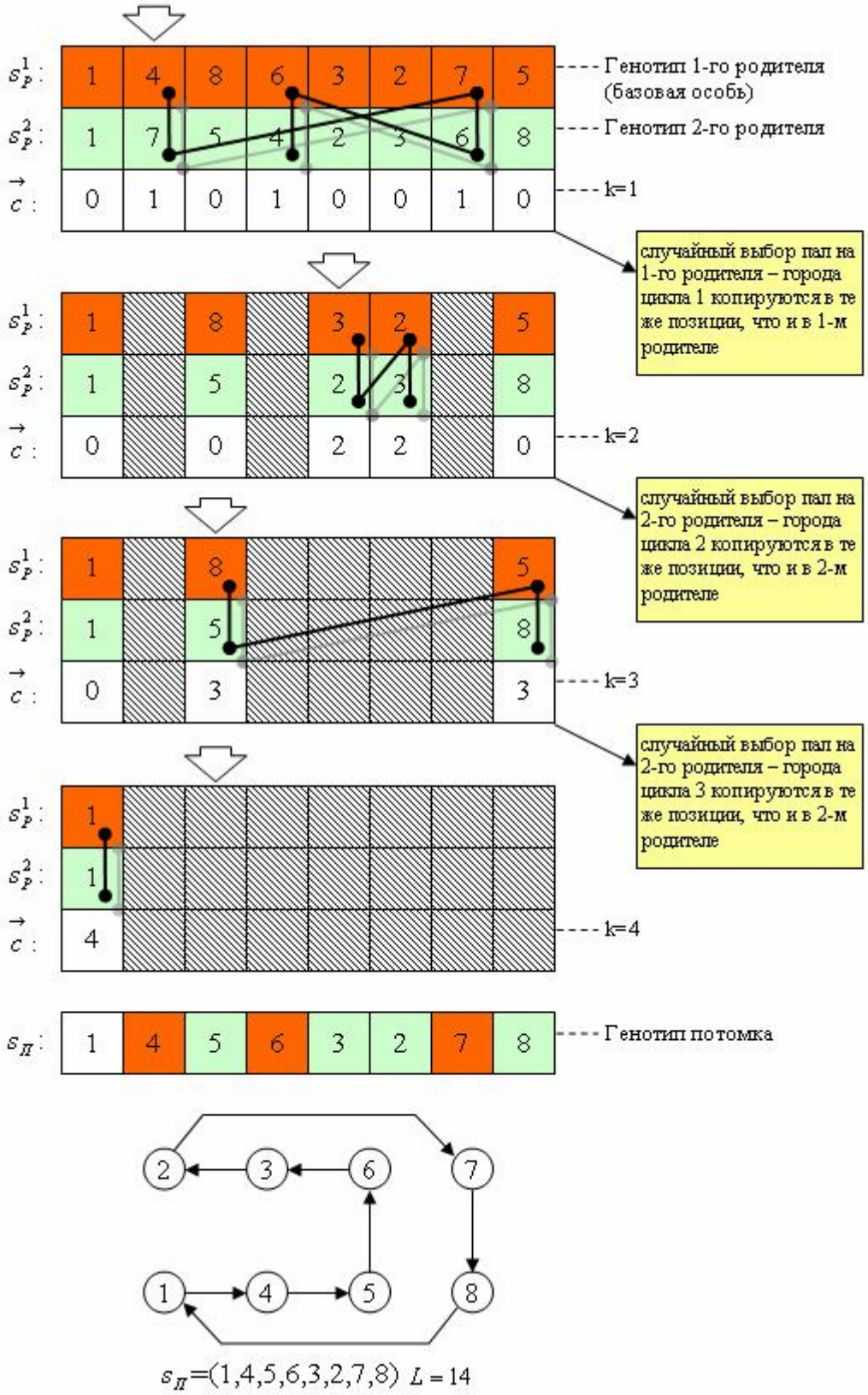

Рис. 2.9. Пример работы циклического кроссовера (см. в приложении задачу 1).

9. Если *j*-го города в s_P^1 еще нет присутствует в *k*-ом цикле ($c_j \neq k$) , то повторяем действия начиная с пункта 6, иначе переходим в пункт 10.
10. Цикл k сформирован. Случайным образом из пары родительских генотипов s_P^1 и s_P^2 выбираем один и копируем напрямую из нее все имена городов из цикла k в генотип потомка $s_П$: $s_П(i) = s_P(i)$ для всех i таких, что $c_i = k$, $i = \overline{1,n}$.
11. Если есть хотя бы один не посещенный город ($\exists i \in \{1,...,n\} : c_i = 0$), то повторяем действия начиная с пункта 4, иначе выход.

Генотип другого потомка формируется аналогичным образом путем перемены ролей «родительских» генотипов. На рисунке 2.9 показан пример работы алгоритма циклического кроссовера.

2.5.3. Частично-отображающий кроссовер

Частично отображающий кроссовер (*partially-mapped crossover*) или *кроссовер PMX* часть генотипа 1-го «родителя» s_P^1 строго отображает в генотип «потомка» генотип потомка $s_П$, а остальная информация о генотипе 2-го «родителя» s_P^2 изменяется после соответствующих операций обмена так, чтобы сохранить порядок и позиции как можно большего числа городов из этого «родителя» в генотипе потомка $s_П$. Используя допустимые генотипы двух «родительских» особей, кроссовер PMX сохраняет допустимость «потомства».

Частный случай кроссовера, основанного на частичном отображении

1. Случайным образом с вероятностью $\frac{1}{n-1}$ из интервала $[1, n-1]$ выбираются две точки кроссовера a_1 и a_2 ($a_1 < a_2$), которые делят «родитель s_P^1 и s_P^2 на три части.
2. Части генотипов s_P^1 и s_P^2, находящиеся между точками кроссовера a_1 и a_2 (последовательность позиций, начиная с (a_1+1) позиции до (a_2-1) включительно будем называть *секциями отображения*:

$C_1 = C(s_P^1)$ – секция отображения первого родителя;

$C_2 = C(s_P^2)$ – секция отображения первого родителя.

3. Элементы секции отображения 1-го родителя полностью копируются в те же самые позиции генотипа потомка $s_П$. При этом в $s_П$ сохраняются порядок, смежность и позиции элементов секции C_1. Остальные элементы в потомке $s_П$ считаются незанятыми.

4. Составляется таблица отображений T элементов $s_P^1(i) \in C_1$ в элементы $s_P^2(i) \in C_2$: $T(s_P^1(i)) = s_P^2(i)$. Для всех остальных элементов $s_P^1(i) \notin C_1$ $T(s_P^1(i)) = s_P^1(i)$.

5. Поочередно перебираются все элементы вне секции отображения 2-го родителя s_P^2, и в те же самые позиции в генотипе потомка $s_П$, если копируемый элемент содержится в секции отображения, то на его место ставится элемент, полученный при помощи таблицы T новые значения: $s_П(i) = T(s_P^2(i))$ для всех $s_P^2(i) \notin C_2$.

Действие последнего шага обеспечивает сохранность порядка и позиций как можно большего числа элементов 2-го родителя. Далее (см. рис. 2.10.) показан пример работы данного алгоритма.

Обобщенный алгоритм частично-отображающего кроссовера

Шаги с 1 по 3 частного случая кроссовера, основанного на частичном отображении.

4. Составляется таблица позиций[1] P_P^2 для генотипа s_P^2, где значение $P_P^1(j)$ указывает на позицию города с именем j в перестановке s_P^2: $P_P^2(s_P^2(i)) = i$ для всех $i = \overline{1,n}$.

[1] Фактически P_P^1 является отображением обратным к s_P^1.

5. Последовательно (слева направо) для всех элементов $s_P^2(i) \in C_2$ и $s_P^2(i) \notin C_1$, где i-текущий номер позиции в секции отображения C_2, выполняется следующая последовательность действий:

   5.1. принимаем $j = i$;

   5.2. рассчитываем позицию j^* из следующей формулы: $j^* = P_P^2(s_P^1(j))$;

   5.3. если позиция в потомке $s_П(j^*)$ уже занята, то принимаем j равным j^* ($j = j^*$) и переходим к пункту 7.1. Иначе переходим к пункту 7.3.

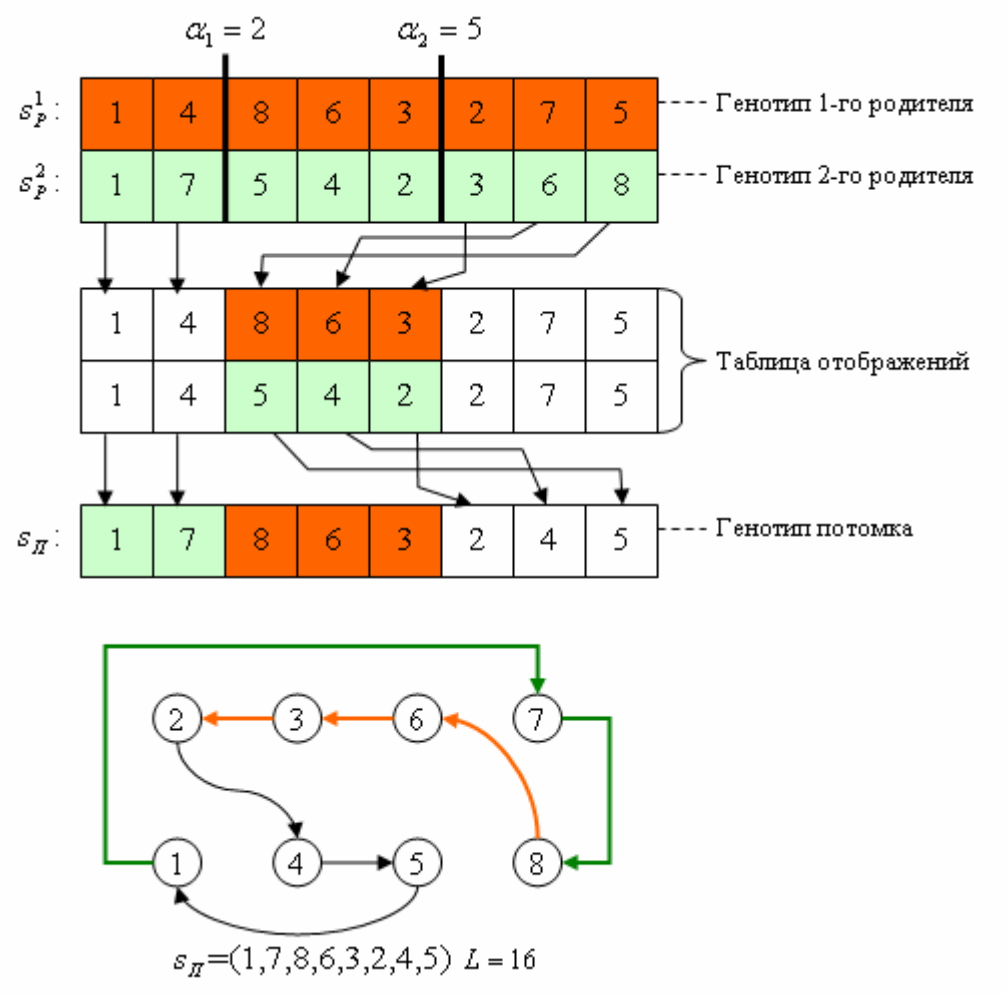

Рис. 2.10. Частного случай кроссовера PMX. Пример работы.

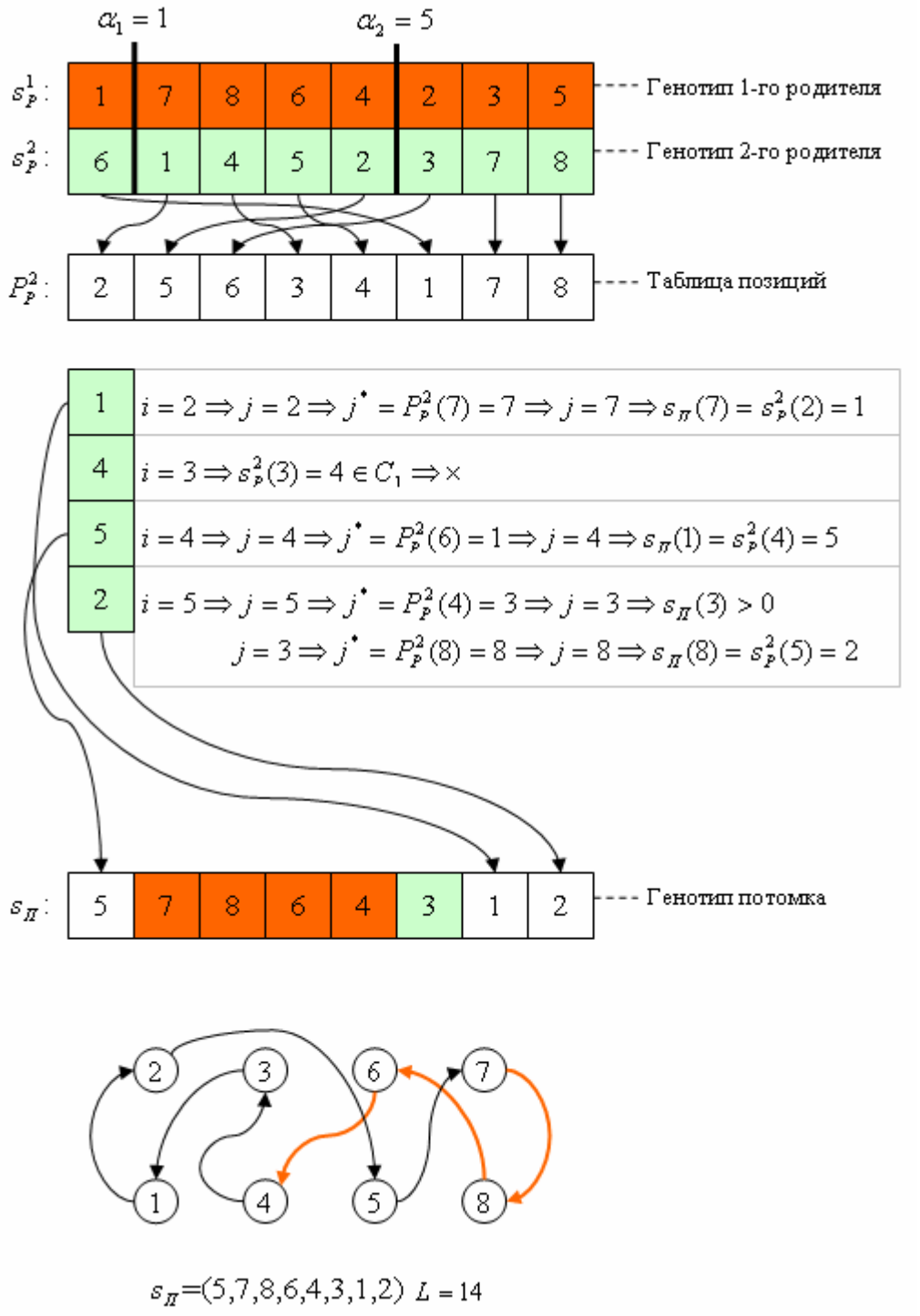

Рис. 2.11. Обобщенный алгоритм кроссовера PMX.
Пример работы..

5.4. Записываем в j-ую позицию потомка значение $s_P^2(i) \in C_2$ и считаем позицию j в потомке занятой: $s_П(j) = s_P^2(i)$.

6. Оставшиеся элементы генотипа 2-го родителя s_P^2 прямо копируются в релевантные незанятые позиции генотипа потомка $s_П$.

Пример работы алгоритма представлен на рисунке (см. рис. 2.11).

2.5.3. Операторы мутации

В качестве операторов мутации для перестановочного кодирования используется модификация классической мутации. Это, например:

- *точечная мутация*: кодировка мутанта получается путем однократной перестановки аллелей в случайно выбранном локусе и соседнем с ним;
- *инверсия*: кодировка c разрывается в двух точках r_1 и r_2 ($r_1 < r_2$), случайно выбранных с равной вероятностью из интервала $[1, (L-1)]$; значения аллелей во всех локусах куска записываются в обратном порядке от точки; кодировка мутанта c^M образуется путем соединения вновь двух старых кусков и нового куска

$$c^M = (c_1, ..., c_{r_1}, c_{r_2}, c_{r_2-1} ... c_{r_1+2}, c_{r_1+1}, c_{r_2+1}, ..., c_L);$$

- *сальтация*: в кодировке c случайным образом выбираются $2k$ ($2k < L$) локусов, в каждой паре локусов переставляются значения аллелей.