§ 1. Общая характеристика методов поиска решений оптимизационных задач

Перед человеком всегда стоит множество задач, которые требуют решения, но ресурсы, имеющиеся в распоряжении, всегда ограничены. Поэтому, как правило, перед тем, как взяться за решение задачи, нам приходиться хотя бы приблизительно определять соотношение между эффективностью ее решения и необходимыми для этого затратами. Можно выделить три основные типа методов поиска оптимальных решений: методы, основанные на математических вычислениях; перечислительные методы и методы, использующие элемент случайности (стохастический поиск).

Методы, основанные на математических вычислениях, изучены наиболее полно. Они могут быть разделены на ненаправленные и направленные методы поиска экстремума.

Суть ненаправленного метода состоит в том, что локальный экстремум ищется путем решения системы уравнений, определяющей необходимые условия существования экстремума. Все полученные решения подвергаются дополнительному анализу с целью выявления истинных экстремумов.

Направленные методы строятся на перемещении от точки к точке в допустимой области, причем направление подобных перемещений связывается с направлением, на которое указывает градиент, и, как правило, некоторыми дополнительными ограничениями.

Обе группы методов давно стали классическими, и их можно рекомендовать для использования в тех задачах, где они применены. К сожалению, круг таких задач достаточно узок. Во-первых, эти методы требуют достаточно гладких целевых функций. Это очень жесткое условие. На практике, особенно в сфере организационного управления, целевые функции могут иметь достаточно произвольный вид, и выполнение условия гладкости можно рассматривать скорее как исключение. Другим, весьма существенным их недостатком является тот факт, что оба метода обеспечивают поиск локальных экстремумов. Поэтому достаточно часто методы, основанные на математических вычислениях, оказываются несостоятельными из-за очень узкой области применимости.

Перечислительные алгоритмы имеют множество видов и форм. Их основная идея состоит в следующем. Как правило, без ограничения общности поисковое пространство любой реальной задачи можно представить в виде конечного множества точек. Поэтому даже те задачи, которые, как предполагается, имеют бесконечное множество допустимых решений, в некотором приближении можно считать дискретными, так для решения всех сколько-нибудь серьезных задач в той или иной мере используется вычислительная техника. Следовательно, чтобы найти оптимальное решение, достаточно просмотреть значения целевой функции во всех этих точках. Для этих целей разработаны различные алгоритмы. Такие методы поиска легко доступны для понимания и прекрасно функционируют при сравнительно небольших размерностях задачи. Но даже такая эффективная схема, как динамическое программирование,

терпит неудачу при увеличении в размерности. Все перечислительные методы становятся крайне неэффективными, когда речь заходит о решении задач большой размерности. А именно, такие задачи встречаются на практике достаточно часто. Причем каждая из них требует какого-либо удовлетворительного решения на базе строго ограниченного объема временных и прочих ресурсов.

Методы стохастического поиска стали активно развиваться сравнительно недавно и имели целью преодолеть ограничения методов первых двух типов. Однако следует признать, что стохастические схемы, которые осуществляют случайный поиск в некоторой окрестности, сохраняя при этом лучшее из просмотренных решений, не обеспечивают требуемую эффективность. Во многих случаях, особенно при решении задач дискретной оптимизации, стохастические алгоритмы не смогли составить достойную конкуренцию перечислительным алгоритмам.

В качестве еще одного из направлений построения методов решения задач дискретной оптимизации, сочетающих идей стохастического поиска и перечислительных алгоритмов, предлагается генетический алгоритм.

Генетические алгоритмы — это алгоритмы поиска, построенные на принципах, сходных с принципами естественного отбора в генетике. Образно говоря, они объединяют в себе принцип выживания наиболее перспективных особей — решений и структурированный обмен информацией, в котором присутствует элемент случайности. Исследователь, поставивший задачу, непосредственно не вмешивается в развивающийся про-

цесс поиска решения, однако, управляет им опосредованно, определяя функцию полезности различных решений.

Впервые идею генетического алгоритма выдвинул Джон Холланд. Внутренняя логика и строгая направленность законов живой природы позволили ему выдвинуть гипотезу о возможности некоторого обобщения генетических процессов, характерных для живой природы, на ряд других наук и, в первую очередь, на прикладную математику. С тех пор генетические алгоритмы активно исследовались, причем использование подходов, основанных на генетических алгоритмах, привело к получению важных результатов как для естественных, так и для искусственных систем.

Генетический алгоритм не является исключительно стохастическим. Он представляет собой пример поисковой процедуры, которая использует случайность как средство осуществления поиска в пространстве параметров. На первый взгляд кажется странным использование случайного выбора в качестве направленного поиска. Использование элемента случайности вовсе не отрицает направленность поиска.

Прежде, чем перейти к детальному анализу генетического алгоритма и выявлению его существенных особенностей, рассмотрим различные подходы к определению целей оптимизации. Произвольный последовательный метод оптимизации обеспечивает переход к лучшему (в соответствии с некоторым критерием) набору значений характеристик задачи и проверке оптимальности (невозможности улучшить) некоторого найденного решения. Иными словами, общая схема оптимизации состоит из двух частей:

1. Улучшения текущего решения;

2. Проверки достижения оптимального решения.

Для многих сложных систем обычно более важно существование технологии улучшения решения. Это связано с тем, что для сложной системы часто требуется какое-либо удовлетворительное решение, а проблема достижения оптимума отходит на второй план. Именно идея последовательного улучшения решений и лежит в основе генетического алгоритма.

Сформулируем основные отличия генетического алгоритма от большинства традиционных алгоритмов решения оптимизационных задач. Таких отличий можно выделить четыре.

1. Генетические алгоритмы работают с кодами, в которых представлен набор параметров, а не с самими параметрами. Обычно параметры представляются строкой конечной длины над конечным алфавитом.

2. Для осуществления поиска генетический алгоритм использует некоторое множество точек поискового пространства, а не осуществляет последовательный переход от точки к точке, как это делается во многих методах иных оптимизаций. На каждой интеграции генетического алгоритма применяется решающее правило, которое и определяет последующую точку. Чтобы избежать проблем, связанных с наличием нескольких локальных экстремумов, генетические алгоритмы строят свою работу на одновременном использовании нескольких начальных точек.

3. Генетические алгоритмы не используют никакой дополнительной информации. Единственное требование — возможность определять допустимость значений па-

раметров и вычислять значение целевой функции в произвольной точке.

4. Генетический алгоритм использует как вероятностные правила для порождения новых точек, так и детерминированные правила для перехода от одного множества к другому.

§ 2. Структура генетического алгоритма

В достаточно общем виде произвольный генетический алгоритм можно формально описать следующим образом.

Инициализация
Создать исходную популяцию.

Основной цикл
WHILE NOT stop DO
BEGIN
Выбрать родителей из популяции;
Создать потомков выбранных родителей;
Подвергнуть воздействию мутации новые особи;
Расширить популяции путем присоединения к ней новых особей;
Сократить расширенную популяцию;
END
Поиск лучшей найденной особи-решения;
END

Рис. 9.2.1. Структура генетического алгоритма

Определение 9.2.1. Поисковое пространство задачи представляется в виде множества *особей.*

Каждая особь представляется символьной строкой, которую часто ассоциируют с хромосомным набором. Генетический алгоритм имеет целью нахождения такой особи, которая обладает наилучшим «генетическим материалом». Качество каждой отдельной особи-решения измеряется при помощи целевой функции.

Определение 9.2.2. Часть поискового пространства, с которой работает генетический алгоритм на конкретном этапе, называется *популяцией.*

В соответствии с рис. 9.2.1, прежде всего, необходимо создать начальную популяцию. После создания начальной популяции измеряется качество каждого из ее элементов. После этого начинается процесс, сходный с процессом эволюции. На каждой интеграции из популяции выбирается m пар родителей.

1. Выбор родителей из популяции

Выбранные из популяции родители производят потомков. Вероятность того, что данный представитель популяции произведет потомство, находится в прямой зависимости от его качества. Например, для популяции объема n i-ый элемент сможет участвовать в производстве потомства с вероятностью:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j}, \qquad (9.2.1)$$

где f_i — значение целевой функции в каждой из n точек популяции.

В реальных алгоритмах могут использоваться и другие, отличные от (9.2.1), формулы. Но обычно в механизме выбора родителей присутствует идея «права сильнейшего», т. е. чем более высокий показатель качества представителя популяции, тем больше у него шансов попасть в список родителей.

Как правило, пара родителей производит пару потомков. Вновь полученные особи подвергаются воздействию мутации. Вероятность мутации достаточно мала, но отлична от нуля. После этого все вновь порожденные особи добавляются к популяции. Затем популяция сокращается до исходного размера путем удаления из нее части особей. Удаление осуществляется согласно критерию селекции, выбранному заранее.

Два основных оператора генетического алгоритма — *оператор порождения нового поколения* и *оператор мутации* – изначально разрабатывались как аналоги процессов, протекающих в естественных системах. Остановимся на каждом из них наиболее подробно.

2. Создание потомков избранных родителей. Оператор скрещивания

Порождение нового поколения состоит из двух этапов: непосредственного воспроизводства (копирования) родительских строк и скрещивания. Простейший опера-

тор скрещивания выполняется за два последовательных шага. Предположим, что скрещиванию подлежат две строки длины n. Равновероятно выбирается натуральное число k от 1 до $n - 1$. Число k называется *точкой разбиения*. В соответствии с ним обе исходные строки разбиваются на две подстроки. Например, первая строка $X_1, \ldots, X_n$ разбивается на подстроки $X_1, \ldots, X_{k-1}$ и $X_k, \ldots, X_n$. Затем строки обмениваются своими подстройками, лежащими после точки разбиения, т. е. элементами с k-го по n-ый. Получаются две новые строки. Схема действия простейшего оператора скрещивания представлена на рис. 9.2.2.

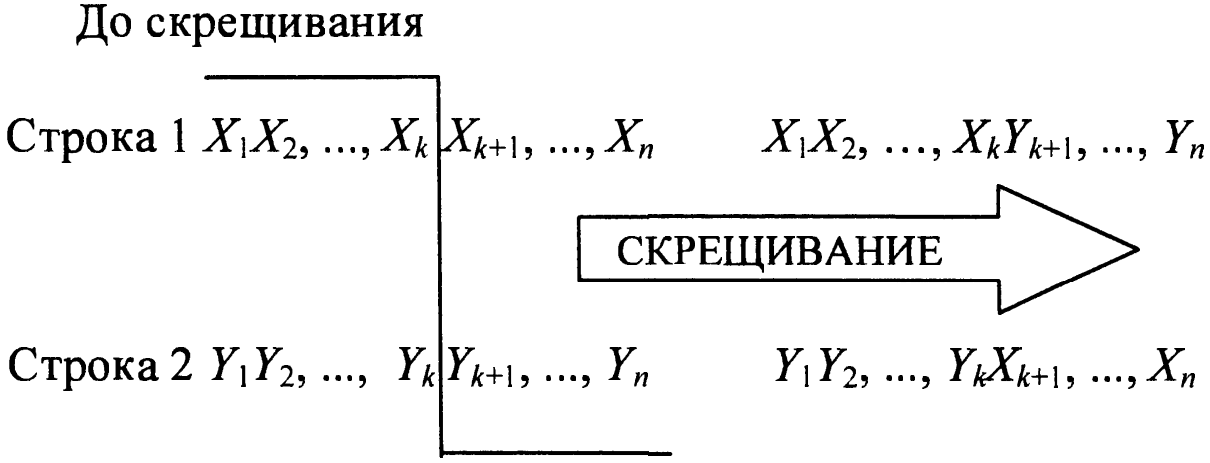

Рис. 9.2.2. Пример оператора скрещивания

3. Оператор мутации

Оператор мутации реализует случайное изменение значения элемента строки. Вероятность мутации обычно выбирается малой. В примерах, которые следуют далее, она составляет около 0,001. Однако этот оператор

играет важную роль, если популяция ущербна в «генетическом» смысле. В этом случае многократное скрещивание по сути не может дать новых результатов, и только оператор мутации приносит обновление информации.

§ 3. Пример генетического алгоритма

Рассмотрим пример построения генетического алгоритма для решения достаточно простой задачи.

Пусть на множестве целых чисел, принадлежащих отрезку [0, 31], задана функция $f(x)=x$. Задача состоит в определении максимума функции $f(x)$.

Для данной задачи выберем двоичное представление точек области определения. Таким образом, алгоритм будет работать с бинарными строками длины 5. Будем следовать схеме, приведенной на рис. 9.2.1.

Создадим исходную популяцию. Пусть объем популяции будет равняться 4. На момент создания исходной популяции мы ничего не можем сказать о «качестве» той или иной особи-строки. Поэтому для инициализации генетического алгоритма логично использовать последовательность реализаций случайной величины, распределенной по закону Бернулли и равновероятно принимающей значение 0 и 1.

На каждом этапе генетического алгоритма будут выбираться две пары родителей, каждая из которых производит по два потомка. Предположим, что в результате инициализации получились строки, как показано в таблице 9.3.1.

Таблица 9.3.1

Пример популяции генетического алгоритма

Строки	*Значения целевой функции*	*Вероятность участия в производстве потомства*
A_1 01011	11	$\frac{11}{43}$
A_2 10010	18	$\frac{18}{43}$
A_3 00010	2	$\frac{2}{43}$
A_4 01100	12	$\frac{12}{43}$

Предположим, что для продолжения потомства были выбраны пары (A_1A_3) и (A_2A_4). Рассмотрим возможные варианты действия операторов скрещивания.

Пусть в первой паре решений точка разбиения попадает между первым и вторым символами, а во второй паре — между третьим и четвертым. Результаты действия оператора скрещивания приведены в таблице 9.3.2.

Таблица 9.3.2

Пример использования оператора скрещивания с одной точкой разбиения

Предки	*Потомки*	*Значения целевой функции*
A_1 0|1011	11011 A'_{12}	27
A_2 1|0010	00010 A''_{12}	2
A_2 100|10	10000 A'_{24}	16
A_4 011|00	01110 A''_{24}	14

Так как вероятность мутации очень мала, то будем считать, что ни один из потомков не подвергался ее воздействию. После двух последних операций: расширения и сокращения, получается обновленная популяция, представленная в таблице 9.3.3.

Таблица 9.3.3

Популяция генетического алгоритма после одной итерации

Строки	*Значение целевой функции*
A'_{12} 11011	27
A_2 10010	18
A'_{24} 10000	16
A''_{24} 01110	14

Заметим, что в рассмотренном примере одна итерация генетического алгоритма привела к заметному улучшению решения, а именно, с 18 по 27 при оптимальном решении 31.

§ 4. Гипотеза «строительных блоков»

Одним из направлений активных исследований последнего времени является гипотеза «строительных блоков». Суть ее состоит в том, что внутри каждой символьной группы, на которой достигается достаточно высокое значение целевой функции, существует фиксированный набор элементов, которые вносят определяющий вклад в значение целевой функции $f(x)$.

Например, если в рассмотренном примере значение 1-го элемента строки равно 1, то значение целевой функции независимо от значений других элементов строки будет лежать в пределах от 16 до 31. Такие подмножества элементов получили названия схем. Каждая схема характеризуется двумя показателями:

- *порядком* — количеством элементов;
- *длиной* — расстоянием от первого символа, входящего в схему, до последнего.

Схемы являются основным средством для анализа эффекта распространения генетического материала и воздействия на него операторов генетического алгоритма.

Рассмотрим произвольную схему H порядка $o(H)$ и длины $\delta(H)$. Предположим, что на t-ом шаге среди представителей популяции $A(t)$ находится $m = m(H,t)$

носителей схемы H. Задача состоит в том, чтобы оценить индивидуальное и совокупное влияние различных операторов, а именно: воспроизводство, скрещивания и мутации, и таким образом построить оценку для количества носителей схемы H в популяции на $(t+1)$-ом шаге.

Первым из операторов генетического алгоритма действует оператор выбора родительских пар для производства потомков. Оценим ожидаемое количество особей-носителей интересующей нас схемы H, отобранных на этом этапе. Обозначим через $f(H)$ среднюю полезность схемы H в момент времени t. Математически это можно записать следующим образом:

$$f(H)=\frac{1}{n_H}\sum_{A_i(t)\,H\subset A_i(t)} f(A_i(t)), \qquad (9.4.1)$$

где n — количество строк-носителей схемы H в популяции.

На этапе выбора строка выбирается в родительскую пару с вероятностью, определяемой по формуле (9.2.1). Будем считать, что на каждой итерации генетического алгоритма выбирается n родителей. Тогда среди строк-родителей ожидаемое количество $m'(H,t+1)$ носителей схемы H составит:

$$m'(H,t+1)=m(H,t)n\frac{f(H)}{\bar{f}}, \qquad (9.4.2)$$

где $\bar{f}$ — усредненное значение функции полезности представителей популяции в момент времени t.

Формула (9.4.2) означает, что частота встречаемости схемы H прямо зависит от отношения ее текущей полезности к средней полезности в целом. Данный закон действует на все схемы без исключения. Поэтому интенсивность роста представительности той или иной схемы прямо зависит от ее полезности. Общая формула для $m'(H,t)$ после выполнения этого этапа следующая:

$$m'(H,t+1) = m(H,t)\left(1+n\frac{f(H)}{\bar{f}}\right). \qquad (9.4.3)$$

Если бы на популяцию действовал только данный оператор, а все представители каждой новой популяции сохранились бы, то в пределе формулу (9.4.3) можно было бы рассматривать как геометрическую прогрессию. В этом случае число носителей любой схемы росло бы экспоненциально.

Очевидно, что копирование строк само по себе не может привести к новым решениям. Эти функции в генетическом алгоритме возложены на операторы скрещивания и мутации.

Рассмотрим строку из приведенного выше примера и две ее схемы:

$$A = 0\ 1\ 0\ 1\ 1$$
$$H_1 = 0\ *\ *\ 1\ *$$
$$H_2 = *\ 1\ 0\ *\ *.$$

Обе схемы как H_1, так и H_2 представлены в строке A и имеют одинаковый порядок, но разную длину. Согласно описанию оператор скрещивания разбивает

родительские строки на две части. Очевидно, что чем больше длина схемы, тем больше вероятность попадания точки разбиения внутрь схемы. Если точка разбиения лежит вне схемы, то данная схема должна выжить при скрещивании.

В нашем примере точка разбиения размещается равновероятно в одной из четырех позиций. Следовательно, схема H_1 будет разрушена с вероятностью ¾, и, наоборот, схема H_2 останется нетронутой с такой же вероятностью. В случае использования простого оператора скрещивания, вероятность выживания схемы при скрещивании p_s определяется из формулы:

$$p_s = 1 - \delta(H) / (l-1),$$

где l — длина строки.

Если же сам оператор скрещивания применяется с вероятностью, например p_c, нижняя оценка для вероятности может быть выражена следующим образом:

$$p_s \geq 1 - p_c \frac{\delta(H)}{l-1}. \qquad (9.4.4)$$

Теперь рассмотрим совокупное влияние на выживание схемы двух рассмотренных операторов: выбора и скрещивания. Будем считать, что операторы выбора родительских пар и скрещивания действуют независимо друг от друга. Основываясь на данном предположении, можно получить оценку:

$$m'(H,t+1) \geq m(H,t)n\frac{f(H)}{\bar{f}}\left[1-p_c\frac{\delta(H)}{l-1}\right]. \qquad (9.4.5)$$

Таким образом, можно сделать вывод, что вероятность сохранения схемы в новом поколении зависит как от ее относительной полезности, так и от ее длины.

Еще один оператор, участвующий в генерации нового поколения, — оператор мутации. Напомним, что *мутация* — это случайное изменение одного из элементов строки, которое происходит с вероятностью p_m. Поэтому для того, чтобы схема H сохранилась, необходимо и достаточно, чтобы все ее $o(H)$ элементов остались без изменения. Это возможно, если строка не подвергается воздействию мутации, либо мутация затрагивает элемент строки, не входящий в данную схему. Следовательно, вероятность того, что мутация не затронет схему составляет $(1-p_m)^{o(H)}$.

Так как обычно значения вероятности мутации малы, то можно считать, что вероятность выживания схемы определяется выражением $1-o(H)p_m$.

$$m'(H,t+1) \geq m(H,t)n\frac{f(H)}{\bar{f}}\left[1-p_c\frac{\delta(H)}{l-1}-o(H)p_m\right]. \qquad (9.4.6)$$

Последняя формула указывает на незначительное влияние фактора мутации. Иногда результат, представленный формулой (9.4.6), называют *фундаментальной теоремой генетического алгоритма*. Поэтому считается, что перспективные, с точки зрения значения целевой

функции, схемы, имеющие малый порядок и малую длину, должны чаще других присутствовать в строках, которые выбираются для воспроизводства потомства, и с большей вероятностью наследоваться потомками.

По мере работы алгоритма эти схемы будут распространяться внутри популяции, а количество их носителей будет возрастать. Похожую картину можно наблюдать и в процессе естественного отбора. С другой стороны, данный факт объясняет перспективность использования генетического алгоритма для решения сложных задач.

Действительно, вместо того чтобы исследовать все возможные комбинации элементов строк, генетический алгоритм строит все более и более хорошие решения, опираясь на лучших представителей предыдущих поколений. В этом случае задача сводится к выявлению наиболее перспективных схем и их последующему комбинированию.

На интуитивном уровне такой подход может показаться простым и очевидным. С формальной точки зрения далеко не все положения гипотезы получили четкое математическое объяснение. Несмотря на то, что уже получены некоторые важные результаты, дискуссия по перспективам построения генетических алгоритмов для решения оптимизационных задач продолжается. Одновременно осуществляются исследования по применению генетических алгоритмов для решения сложных задач комбинаторной оптимизации. Одной из таких задач является задача коммивояжера.

§ 5. Генетический алгоритм для задачи коммивояжера

Задача коммивояжера — это классическая *NP*-полная задача, к которой сводится решение многих других задач.

Пусть дано множество, состоящее из n городов и матрица расстояний между ними. Задача состоит в том, чтобы определить кратчайший путь, проходящий через все города. При этом каждый город посещается только один раз, а сам путь завершается в том же городе, откуда он начинался. Решением задачи является перестановка длины n, отражающая последовательность посещения городов. Качеством решения принято считать длину пути.

Для решения задачи коммивояжера не могут быть использованы те же операторы скрещивания и мутации, что и в простейшем случае, описанном в § 3. В данном случае каждый из операторов должен обеспечивать переход от перестановки к перестановке, что определяет особую специфику операторов генетического алгоритма.

1. Выбор родителей для популяции

Цель оператора скрещивания заключается в производстве вариантов новых решений на основании информации, заложенной в родителях, участвующих в создании потомков. Операторы скрещивания можно условно разделить на две группы по способу построения новых решений: детерминированные и вероятностные.

Детерминированный подход предполагает наличие жестко заданных правил построения новых решений,

при этом для каждой пары родительских решений однозначно определена пара решений-потомков.

Вероятностные операторы скрещивания предполагают использование некоторой вероятностной модели, и как следствие, внесение элемента случайности при построении новых решений. В этом варианте для каждой пары родительских решений можно говорить только о распределении возможных пар решений-потомков. Наиболее распространенными вероятностными операторами скрещивания стали операторы, использующие разбиение исходных решений на фрагменты, подлежащие обмену, и операторы, использующие принцип «маскирования» генов. Представители каждого из описанных классов операторов генетического алгоритма представлены ниже.

Вариант 1. Оператор скрещивания с использованием точек разбиения.

Проиллюстрируем порядок действия оператора конкретным примером для двух подстановок длины 5: (12345) и (34521).

1. Равновероятно выбираются точки разрыва подстановок. Пусть, например, первая точка разрыва будет лежать между первым и вторым элементами, а вторая — между четвертым и пятым. Получаем:

(1|234|5) и (3|452|1)

2. Подстановки обмениваются выделенными фрагментами. Остальные элементы перестановок заменяются на символы * (звездочка). В результате выполнения данной операции получаем:

(*|452|*) и (*|234|*)

3. Символы * (звездочка) в каждом из шаблонов заменяется на цифры в порядке их следования в соответствующей перестановке, начиная со второй цифры выделенного фрагмента. Если просматриваемая цифра уже присутствует в перестановке, то она пропускается. Если достигается конец перестановки, то алгоритм переходит на ее начало и продолжает работу. В результате получаются две новые перестановки:

(14523) и (52341)

Оператор, построенный в соответствии с представленной схемой, поддерживает гипотезу «строительных блоков». В качестве строительных блоков рассматриваются фрагменты пути — подпоследовательности посещения городов, которые обладают хорошими, с точки зрения используемого критерия, характеристиками.

Вариант 2. Оператор скрещивания с использованием слияния перестановок.

Для варианта, когда применяется слияние перестановок, алгоритм построения двух новых решений носит детерминированный характер. Сначала рассматриваются первые элементы родительских решений, которые в том же порядке заносятся в решение-потомка. Затем рассматривается следующая пара элементов и т. д. При этом повторяющиеся элементы пропускаются в связи с ограничением, накладываемым существом понятия перестановки. Для построения второго решения-потомка

используется такой же алгоритм, но родительские строки рассматриваются в обратной последовательности.

Рассмотрим пример. Пусть дана пара перестановок (12345678) и (37516824). Тогда после первой итерации новые решения будут иметь вид (13******) и (31******). После второй — (1327****) и (3172****). Окончательный результат — (13275468) и (31725468). Данный оператор может служить примером использования строго детерминированного механизма построения решений-потомков.

Вариант 3. Оператор скрещивания с использованием «маски».

Одним из типовых подходов к конструированию операторов скрещивания генетического алгоритма является использование различных алгоритмов, основанных на маскировании элементов решений. Для задачи коммивояжера действие маски распространяется на перестановки. Пусть даны две перестановки длины n. Тогда генерируется последовательность бит длины n, где вероятность появления 1 составляет p_c.

В обеих исходных перестановках элементы, соответствующие нулям битовой маски, остаются на своих местах. Остальные элементы заполняются в том порядке, в котором они встречаются во втором решении. Например, даны две перестановки (12345678) и (37516824). Вероятность появления единицы p_c равна 0,5, и сгенерирована битовая последовательность (01110100). Тогда в соответствии с описанным алгоритмом после наложения маски получается два новых решения-потомка (1***4*78) и (3***1*24). Окончательный вид новых решений: (13564278) и (35671824).

2. Оператор мутации

Любая подстановка, полученная при помощи оператора скрещивания, подвергается воздействию мутации. Существуют различные подходы к заданию оператора мутации. Среди операторов мутации различают *инверсные* операторы и операторы *замены*. При использовании инверсных операторов части вновь полученных решений инвертируются. При использовании операторов замены один из элементов с некоторой вероятностью заменяется на другой. В случае, когда решениями являются перестановки, оператор замены представляет собой транспозицию двух элементов. Кроме того, специфические ограничения на вид решений ЗК привели к построению некоторых новых операторов мутации, ориентированных на решение только данной задачи.

Вариант 1. Оператор мутации с заменой.

Оператор равновероятно выбирает пару элементов перестановки и осуществляет их транспозицию.

Вариант 2. Инверсный оператор мутации.

В перестановке равновероятно выбирается пара разделителей и фрагмент перестановки, заключенный между разделителями, которая инвертируется. Например, если для перестановки (12|34567|8) выбраны разделители, первый из которых находится между вторым и третьим элементами, а второй — между седьмым и восьмым, то в результате применения инверсного оператора мутации будет получена перестановка (12765438).

Вариант 3. Оператор мутации с использованием сдвига.

Внутри перестановки равновероятно выбираются два разделителя, и осуществляется циклический сдвиг

перестановки. Для определенности можно рассматривать циклический сдвиг влево. Применение данного оператора мутации к рассмотренной выше перестановке при тех же условиях приведет к получению перестановки (12734568).

В научной литературе представлены и другие возможности задания операторов скрещивания и мутации, вид которых, как правило, определяется спецификой каждой конкретной решаемой задачи. Однако даже в случае достаточно сложного пространства поиска существуют аналоги описанных операторов генетического алгоритма, и их использование позволяет достичь приемлемого результата.

Пример 9.5.1. Решение задачи коммивояжера генетическим алгоритмом.

Пусть требуется решить задачу коммивояжера для пяти городов с исходной матрицей:

$$\begin{pmatrix} \infty & 4 & 6 & 2 & 9 \\ 4 & \infty & 3 & 2 & 9 \\ 6 & 3 & \infty & 5 & 9 \\ 2 & 2 & 5 & \infty & 8 \\ 9 & 9 & 9 & 8 & \infty \end{pmatrix}$$

Данную задачу можно решить каким-либо из широко известных алгоритмов, например, с помощью алгоритма Литтла (см. § 3 гл. 6). Оптимальное решение задачи $5 \to 1 \to 4 \to 2 \to 3 \to 5$ имеет стоимость 25.

Зная точное решение поставленной задачи, попытаемся применить генетический алгоритм. Будем считать, что в генетическом алгоритме используются операторы скрещивания и мутации, заданные в соответствии с вариантом 1.

Создадим случайным образом начальную популяцию. Пусть получены четыре перестановки, представленные в таблице 9.5.1.

Таблица 9.5.1

Начальная популяция генетического алгоритма для решения задачи коммивояжера

Перестановка	*Стоимость пути*
A_1 (12345)	29
A_2 (21435)	27
A_3 (54312)	33
A_4 (43125)	33

Заметим, что различные перестановки A_3 и A_4 на самом деле являются различными представлениями одного и того же решения. Тем не менее, это не означает, что при скрещивании с другими перестановками они поведут себя одинаковым образом.

Предположим далее, что для производства потомков были выбраны пары перестановок A_1 и A_3, A_2 и A_4. Схема одного из возможных вариантов появления потомства приводится ниже.

A_1 (1|23|45) $\xrightarrow{\text{скрещивание}}$ (*|43|**) $\longrightarrow$ (14325)

A_3 (5|43|12) (*|23|**) (52341) $\xrightarrow{\text{мутация}}$ (53241)

A_2 (2|143|5) $\xrightarrow{\text{скрещивание}}$ (*|312|*) $\longrightarrow$ (53124)

A_4 (4|312|5) (*|143|*) (51432)

После расширения и усечения популяции новая популяция будет иметь вид, представленный в таблице 9.5.2.

Таблица 9.5.2

Популяция генетического алгоритма после одной итерации

Перестановка	*Стоимость пути*
A_1' (12345)	29
A_2' (21435)	27
A_3' (14325)	28
A_4' (51432)	28

Предположим, что на втором этапе для воспроизводства были выбраны пары перестановок A_1' и A_4', A_2' и A_3'. Аналогично первой итерации схема появления потомства приводится ниже.

A_1' (123|45) $\xrightarrow{\text{скрещивание}}$ (514|**) $\longrightarrow$ (51423) *оптимальное решение*

A_4' (514|32) (123|**) (12354)

A_2' (|21|435) $\xrightarrow{\text{скрещивание}}$ (|14|**) $\longrightarrow$ (14235) *оптимальное решение*

A_1' (|14|325) (|21|**) (21435)

В рассмотренном примере генетический алгоритм на первом шаге улучшил качество популяции, а на втором — достиг оптимального решения.

Рассмотрим первую пару, давшую оптимальное решение. Один из родителей является носителем элемента оптимального пути: $5 \to 1 \to 4$. Вторую часть $2 \to 3$ потомок унаследовал от первого родителя, так как в первой перестановке число 2 встречается раньше числа 3. Таким образом, путь, составленный из этих двух частей, оказался оптимальным.

Проиллюстрировав принцип действия генетического алгоритма, рассмотрим эффективность его применения к задачам, имеющим существенно большую размерность.

Например, необходимо решить задачу коммивояжера размерности 100. Как правило, при решении задач такой размерности традиционные алгоритмы требуют достаточно большого времени для поиска оптимального решения.

Предположим для большей наглядности, что города расположены в вершинах выпуклого многоугольника. Отметим, что такая дополнительная информация не привносится в описанную схему алгоритма. Легко определить, что для данной задачи оптимальным является любое решение σ, при котором города обходятся последовательно по циклу. Одним из таких решений является тождественная перестановка. Ясно, что стоимость оптимального решения равна периметру многоугольника.

Эксперимент состоит в применении генетического алгоритма к данной задаче. При этом параметры алгоритма: объем популяции и число итераций, модифици-

руется. Результатом каждого эксперимента является наилучшее решение, полученное в ходе работы алгоритма.

В верхней строке таблицы приведены размерности популяций, для которых проводилась инициализация алгоритма. В левой колонке — количество проведенных итераций. Стоимость оптимального решения для исследованной задачи равнялась 100.

Эксперименты показали, что на практике сходимость генетического алгоритма сильно зависит от значений его параметров. Для рассмотренной задачи критической оказалась размерность популяции 100. Экспериментальные результаты приведены в таблице 9.5.3.

Таблица 9.5.3

Результаты численных экспериментов
для задачи коммивояжера

	20	*50*	*100*	*150*
100	2224	1546	1320	884
200	1820	1448	884	528
500	1756	1326	336	202
1000	1534	1104	158	108
1500	1478	944	112	100
2000	926	712	100	100
5000	882	396	100	100

Литература

1. *Ахо А.*, *Хопкрофт Дж.*, *Ульман Дж.* Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979.
2. *Берж К.* Теория графов и ее применение. — М.: ИЛ, 1962.
3. *Васильев Ф. П.* Численные методы решения экстремальных задач. — М.: Наука, 1980.
4. *Гудман С.*, *Хидетниеми С.* Введение в разработку и анализ алгоритмов. — М.: Мир, 1981.
5. *Гэри М. Р.*, *Джонсон В. С.* Труднорешаемые задачи и вычислительные машины. — М.: Мир, 1983.
6. *Емеличев В. А.*, *Ковалев М. М.*, *Кравцов М. К.* Многогранники, графы, оптимизация. — М.: Наука. 1981.
7. *Зыков А. А.* Основы теории графов. — М.: Наука, 1987.
8. *Конвей Р. В.*, *Максвелл В. Л.*, *Миллер Л. В.* Теория расписаний. — М.: Наука, 1975.
9. *Кормен Т.*, *Лейзерзон Ч.*, *Ривест Р.* Алгоритмы: построение и анализ. — М.: МЦНМО, 1999.
10. *Колобашкин С. М.*, *Коваленко А. П.*, *Смирнов С. Н.* Методы дискретной оптимизации. – М.: Изд. в/ч 33965, 1990.
11. *Кофман А.* Введение в прикладную комбинаторику. — М.: Наука, 1975.
12. *Кристофидес Н.* Теория графов. Алгоритмический подход. — М.: Мир, 1978.
13. *Оре О.* Теория графов. — М.: Наука, 1980.