КУРС ЛЕКЦИЙ

Часть 2

«ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ»

Содержание

Введение

В настоящее время развивается новое направление в теории и практике искусственного интеллекта – *эволюционные вычисления*. Этот термин обычно используется для общего описания алгоритмов поиска, оптимизации или обучения, основанных на некоторых формализованных принципах естественного эволюционного отбора. Особенности идей эволюции и самоорганизации заключаются в том, что они находят подтверждение не только для биологических систем развивающихся много миллиардов лет. Эти идеи в настоящее время с успехом используются при разработке многих технических и, в особенности, программных систем. Эволюционные вычисления используют различные модели процесса эволюции. Среди них можно выделить следующие основные парадигмы:

1) генетические алгоритмы (ГА);
2) эволюционные стратегии (ЭС);
3) эволюционное прогнозирование (ЭП);
4) генетическое программирование (ГП).

Они отличаются друг от друга способами представления искомых решений и различным набором операторов, используемых в процессе моделирования эволюции.

Эволюционный поиск – это последовательное преобразование одного конечного множества промежуточных решений в другое. Основой для его возникновения считается модель биологической эволюции и методы случайного поиска. Эволюционный процесс представляется как способность “лучших” хромосом оказывать большее влияние на состав новой популяции на основе длительного выживания и более многочисленного потомства. Само преобразование можно назвать алгоритмом поиска, или алгоритмом эволюции. Выделяют три особенности алгоритма эволюции:

1) каждая новая популяция, состоит только из «жизнеспособных» хромосом;
2) каждая новая популяция «лучше» (в смысле близости к решению поставленной задачи) предыдущей;

3) в процессе эволюции последующая популяция зависит только от предыдущей.

1. Определение и основные свойства генетических алгоритмов

В конце 1960-х годов американский исследователь Джон Холланд предложил в качестве механизма комбинаторного перебора вариантов решения оптимизационных задач использовать методы и модели механизмов развития (эволюционирования) органического мира на Земле. Поскольку основные законы эволюции живых организмов были впервые исследованы и описаны генетикой, то и предложенный подход получил название *генетические алгоритмы*.

ГА есть случайно направленные поисковые алгоритмы, основанные на механизмах естественной селекции и натуральной генетики. Они реализуют принцип «выживание сильнейших» среди рассматриваемых структур, формируя и изменяя поисковый алгоритм на основе моделирования эволюции. Они основаны на следующих механизмах естественной эволюции:

1) Первый принцип основан на концепции выживания сильнейших и естественного отбора по Дарвину, который был, сформулирован им в 1859 году в книге «Происхождение видов путем естественного отбора». Согласно Дарвину особи, которые лучше способны решать задачи в своей среде, выживают и больше размножаются (репродуцируют). В генетических алгоритмах каждая особь представляет собой решение некоторой проблемы. По аналогии с этим принципом особи, лучшие в смысле решения поставленной задачи, имеют большие шансы выжить и репродуцировать. Формализация этого принципа, как мы увидим далее, дает оператор репродукции (селекция и редукция).
2) Второй принцип обусловлен тем фактом, что хромосома потомка состоит из частей полученных из хромосом родителей. Этот принцип был открыт в 1865 году Менделем. Его формализация дает основу для оператора скрещивания.

3) Третий принцип основан на концепции мутации, открытой в 1900 году де Вре. Первоначально этот термин использовался для описания существенных (разных) изменений свойств потомков и приобретение ими свойств, отсутствующих у родителей. По аналогии с этим принципом генетические алгоритмы используют подобный механизм для изменения свойств потомков и тем самым повышая разнообразие (изменчивость) особей в популяции (множество решений).

Эти три принципа составляют ядро ГА. Используя их, популяция (множество решений данной проблемы) эволюционирует от поколения к поколению.

1.1 Общая структура ГА. Основные операторы ГА.

В ГА из всего пространства поиска выделяется некоторое множество точек этого пространства (потенциальных решений), которое в терминах натуральной селекции и генетики называется *популяцией*. Каждая *особь* популяции – потенциальное решение задачи, представляется *хромосомой* – структурой элементов – *генов*. В свою очередь, произвольный ген хромосомы принимает значение – *аллель*, из некоторого алфавита, задающего кодовое представление точек пространства поиска решений. В простейшем случае особью может быть двоичная строка. ГА работает с кодированными структурами безотносительно их смысловой интерпретации. В этом случае сам код и его структура описываются понятием *генотип*, а его интерпретация, с точки зрения решаемой задачи, понятием *фенотип*. На множестве решений определяется целевая (fitness) функция (ЦФ), которая позволяет оценить близость каждой особи к оптимальному решению – способность к выживанию. Генетический алгоритм поиска решения заключается в моделировании эволюционирования подобной искусственной популяции. Популяция развивается (эволюционирует) от одного поколения к другому. Создание новых особей в процессе работы алгоритма происходит на основе моделирования процесса размножения. При этом особи-решения, участвующие в процессе воспроизводства называют родителями, а получившиеся в результате особи-

решения – потомками. В каждом поколении множество особей-потомков создается, используя части особей-родителей и добавляя новые части с «хорошими свойствами».

Непосредственная генерация новых кодовых строк из двух выбранных происходит за счет применения оператора кроссинговера (скрещивания). Моделирование процесса мутации новых особей осуществляется за счет применения оператора мутации, которые позволяют внести в фенотип потомка "хорошие свойства" путем некоторых изменений его генотипа. В основном, вероятность применения оператора мутации имеет небольшое значение << 0,1.

Являясь разновидностью методов случайного поиска, генетические алгоритмы позволяют найти хорошие (квазиоптимальные), но не оптимальные решения. Основное их преимущество состоит в том, что они позволяют найти более "хорошие" решения очень трудных задач за меньшее время, чем другие алгоритмы. Отрицательным свойством эволюционных, в том числе и генетических методов является то, что они требуют адаптации к каждому конкретному классу задач путем выбора определенных характеристик и параметров. К таким параметрам относятся, прежде всего: размер популяции, метод кодирования решения, набор генетических операторов, задействованных в каждом алгоритме и вероятности их использования, критерий остановки процесса поиска.

Критерием остановки может быть, во-первых, прохождение определенного числа поколений, во-вторых, достижение требуемого качества решений, и, наконец, в-третьих, отсутствие улучшения качества решений из-за вырождения текущей популяции. Общая схема генетического алгоритма представлена на рисунке 1.1

ГА позволяют решать задачи прогнозирования, классификации, поиска оптимальных вариантов, и совершенно незаменимы в тех случаях, когда в обычных условиях решение задачи основано на интуиции или опыте, а не на строгом (в математическом смысле) ее описании.

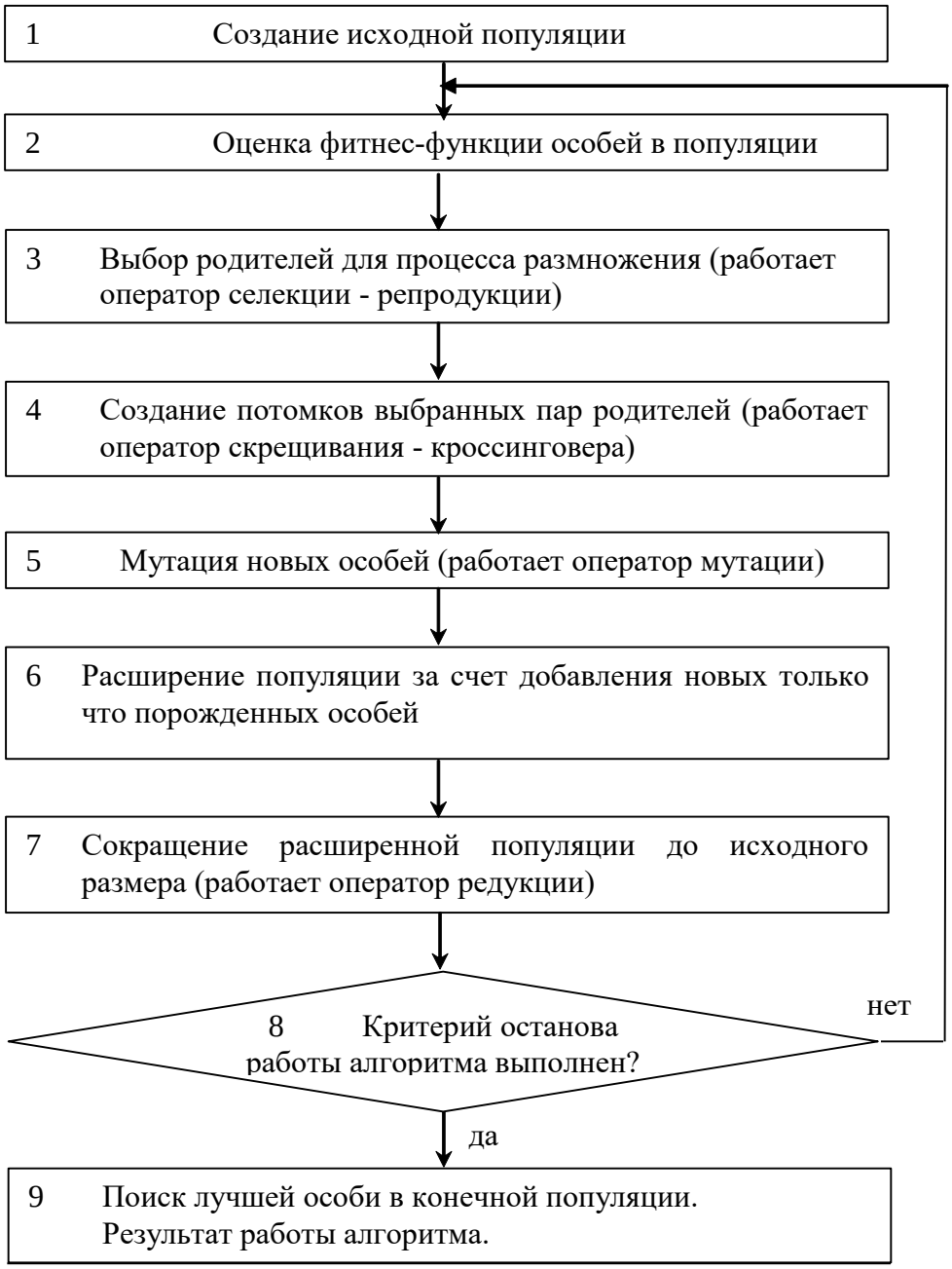

Рис. 1.1 Схема классического «простого» генетического алгоритма

При разработке модели ГА исследователи преследуют двоякую цель:

1) абстрактно и формально объяснить процессы адаптации (приспосабливаемости) в естественных системах;

2) спроектировать искусственные программные и аппаратные системы, которые содержат механизмы естественных систем.

Основные отличия ГА от других оптимизационных и поисковых процедур заключаются в следующем:

1) поиск субоптимального решения основан на оптимизации случайного множества решений с различными оценками, а не одного решения;

2) в аспекте преобразования решение рассматривается как некоторая закодированная структура, а не как совокупность параметров. Это позволяет в некоторых случаях ускорить процесс обработки данных и, как следствие, скорость поиска;

3) для оценки пригодности решения наряду с использованием целевой функции дополнительно моделируются правила выживания в исследуемом множестве. Эти правила повышают разнообразие множества решений, которое необходимо для выполнения пункта 1);

4) при инициализации, преобразовании и других видах обработки решения широко используются вероятностные правила, которые вносят в направленность генетического поиска элементы случайности. Тем самым решается проблема преодоления барьеров локальных оптимумов.

1.2 Простой генетический алгоритм

Простой ГА впервые был описан Д.Голдбергом [2] на основе работ Д.Холланда [3]. Предварительно простой ГА случайно генерирует популяцию последовательностей-стрингов (хромосом). Затем алгоритм генерирует новые популяции, используя как отправную точку начальную популяцию и применяя множество основных генетических операторов:

1) оператор репродукции (ОР);
2) оператор кроссинговера (скрещивания, ОК) ;
3) оператор мутации (ОМ).

Рассмотрим работу простого ГА на следующем примере, приводимом в известной монографии Д.Голдберга [2]. Нужно найти максимальное значение функции на множестве целочисленных (для простоты) аргументов из отрезка [0,31] (Рис.1.2). Для объяснения простого ГА построим следующую таблицу 1.1.

Здесь хромосомы (решения) столбца 2 сгенерированны случайным образом. Суммарное значение ЦФ на четырёх хромосомах этой таблицы равно 1170. Значения ЦФ для каждой хромосомы определяется в соответствии с

максимизируемой функцией $f(x) = x^2$ как возведённое в квадрат десятичное значение двоичного кода хромосом.

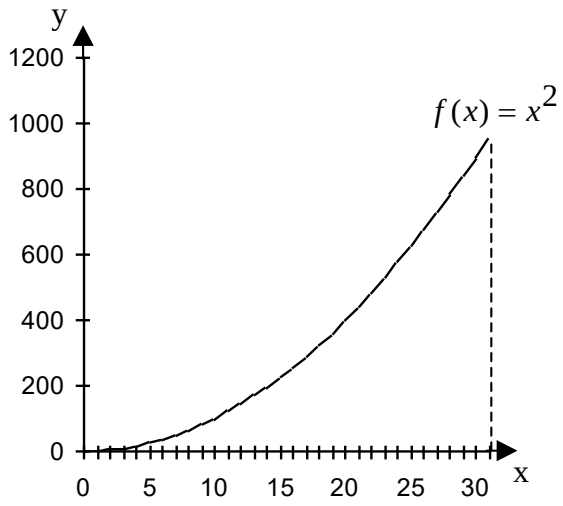

Рис.1.2 График функции $f(x) = x^2$

Репродукция Таблица 1.1

№ хромо-сомы	Начальная популяция особей	Десятичное значение x	Значение ЦФ f(x)	Нормализо-ванная величина $\frac{f(x_i)}{\sum f(x_j)}$	Ожидаемое количество копий хромосом после репродукции	Реальное число копий хромосом после репродукции
1	01101	13	169	0,14	0,56	1
2	11000	24	576	0,49	1,97	2
3	01000	8	64	0,06	0,22	0
4	10011	19	361	0,31	1,23	1
Суммарная ЦФ f(x)			1170	1,00	4,00	4
Среднее значение ЦФ $\overline{f(x)}$			293	0,25	1,00	1
Максимальное значение ЦФ f(x)			576	0,49	1,97	2

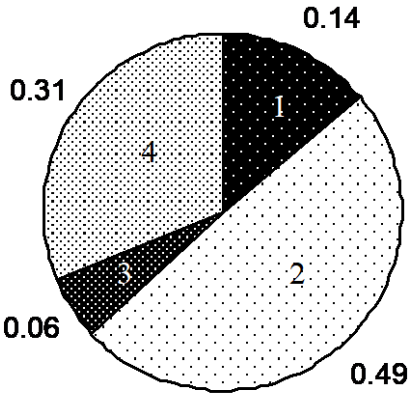

Рис.1.3 Модель ОР в виде асимметричного взвешенного колеса рулетки

Кроссинговер Таблица 1.2

№ хромосомы	Промежуточная популяция после репродукции	Пары хромосом (выбраны случайно)	Точки ОК (k)	Промежуточная популяция после скрещивания	Значение x	Значение f(x)
1	0 1 1 0 | 1	1-2	4	01100	12	144
2	1 1 0 0 | 0	1-2	4	11001	25	625
3	1 1 | 0 0 0	3-4	2	11011	27	729
4	1 0 | 0 1 1	3-4	2	10000	16	256
Sum f(x)		1754				
Среднее значение ЦФ $\overline{f(x)}$		439				
Максимальное значение ЦФ f(x)		729				

Репродукция – процесс, в котором хромосомы копируются согласно значениям их ЦФ. Биологи называют эту функцию – fitness (функция годности, адекватности или соответствия). При этом в процессе копирования хромосомы с “лучшим” значением ЦФ имеют большую вероятность попадания одного или более потомков в следующее поколение. Очевидно, что оператор репродукции (ОР) является искусственной версией натуральной селекции, “выживания сильнейших” по Дарвину. Этот оператор представляется в алгоритмической форме различными способами. Самый простой – построение асимметричного

колеса рулетки, в котором каждая хромосома имеет поле, пропорциональное значению его ЦФ.

Для репродукции хромосом используем модель асимметричного взвешенного колеса рулетки (Рис.1.3). На рисунке поля колеса рулетки соответствует значению ЦФ в процентах по отношению к суммарному значению ЦФ (столбец 5, Табл.1.1). Выбор хромосомы моделируется вращением колеса рулетки. После её останова указатель определяет хромосому, выбранную для следующего оператора (в имитационном моделировании генерируется заданное дискретное вероятностное распределение). Очевидно, что хромосома, которой соответствует больший сектор рулетки, имеет большую вероятность попасть в следующее поколение и породить большее количество потомков. В результате выполнения ОР формируется промежуточная популяция, хромосомы которой будут использованы для построения популяции следующего поколения с помощью оператора кроссинговера (ОК).

Выбираем хромосомы для промежуточной популяции, вращая колесо рулетки 4 раза, что соответствует мощности начальной популяции. Для выполнения ОР используем значения столбцов 5-7 таблицы 1.1. Величину $f(x_i)/\sum f(x)$ называют вероятностью выбора копий (хромосом) при ОР и обозначают

$$P_i(OP) = f(x_i)/\sum f(x). \quad (1)$$

Тогда ожидаемое число копий *i*-ой хромосомы после ОР определяется так:

$$N_i = P_i(\text{ОР}) * n, \qquad (2)$$

где n – число хромосом в популяции.

Число копий хромосомы, переходящих в следующее поколение, часто также определяют на основе выражения

$$\tilde{N}_i = f(x_i) / \bar{f}(x), \qquad (3)$$

где $\bar{f}(x)$ - среднее значение ЦФ по текущей популяции.

Округленные данные приложения формулы (1) приведены в 5-ом столбце, а формул (2), (3) в 6-ом столбце таблицы 1.1. Как видно из последнего (7-го столбца) в результате хромосомы 1 и 4 получают 1 копию, хромосома 2 – 2 копии, и хромосома 3 – не получает копий. Т.е. «лучшие» хромосомы дали большее число копий, «средние» – остаются, «плохие» – умирают после ОР. В результате выполнения ОР формируется промежуточная популяция (столбец 2, табл.1.2)

По Холланду [3] одноточечный или простой ОК выполняются в три этапа:

1-й этап. Две хромосомы

Точка кроссинговера

$$A = a_1 a_2 \ldots a_k \,|\, a_{k+1} \ldots a_L$$
$$B = b_1 b_2 \ldots b_k \,|\, b_{k+1} \ldots b_L$$

выбираются случайно из промежуточной популяции, сформированной при помощи оператора репродукции (ОР). Иногда 1-й этап может выполняться и без ОР.

2-й этап. Случайно выбирается число $k \in [1, L-1]$.

3-й этап. Две новых хромосомы формируются из А и В путем обмена подстроками

$$A' = a_1 a_2 \ldots a_k \;\; b_{k+1} \ldots b_L$$
$$B' = b_1 b_2 \ldots b_k \;\; a_{k+1} \ldots a_L$$

До применения кроссинговера

: 0110| 1 ↕

: 1100| 0

⇓

После применения кроссинговера

1’ : 0110| 0

2’ : 1100| 1

Рис.1.4 Пример одноточечного кроссинговера

Например, рассмотрим кроссинговер хромосом 1 и 2 из промежуточной популяции из таблицы 1.2 (Рис.1.4). Выбираем случайным образом число $1 \le k \le L-1 = 4$. Далее 2 новых хромосомы создаются, меняя местами части хромосом до или после точки кроссинговера. Пусть K=4, тогда имеем процесс на рис.1.4. Хромосомы (1,2) называют родителями, а (1',2') – потомками.

Механизм ОР и ОК удивительно прост. Он выполняет случайную генерацию чисел, копирование хромосом и частичный обмен подстрок хромосом. Удивительно как два таких простых оператора могут образовывать мощный поисковый механизм.

Продолжим пример для $f(x) = x^2$. Получаем следующую таблицу 1.2. В ней во 2-м столбце содержится промежуточная популяция, полученная в результате репродукции, а в 5-м – новая популяция, полученная в результате применения кроссинговера к парам хромосом 1-2 и 3-4 промежуточной популяции.

Сравнение таблиц 1.1 и 1.2 показывает, что после одной генерации улучшились все показатели популяции: среднее и максимальное значения ЦФ.

Далее, согласно схеме классического простого ГА, выполняется оператор мутации (ОМ). Этот оператор играет вторичную роль в ПГА. Обычно вероятность мутации P_M очень невелика.

Мутация Таблица 1.3

№ хромосомы	Популяция после кроссинговера	Новая популяция после мутации	Значения *x*	Значения *y*
1	01100	01100	12	144
2	11001	11001	25	625
3	11011	11111 - мутация	27 31	961
4	10000	10000	16	256
$Sumf_i(x)$			1986	
Среднее значение ЦФ f(x)			496,5	
Максимальное значение ЦФ			961	

Оператор мутации (ОМ) выполняется в 2 этапа:

1-й этап. В хромосоме A=$a_1a_2...a_L$ случайно выбирается позиция $1 \le k \le L$

2-й этап. Производится инверсия значения гена в k-й позиции

$a'_k = \bar{a}_k$

Применим ,например, ОМ к хромосоме 3 с ЦФ = 27 (столбец 2, табл.1.3). Выберем позиции 3 и после ОМ и получим:

3 : 11011 => 3' : 11111

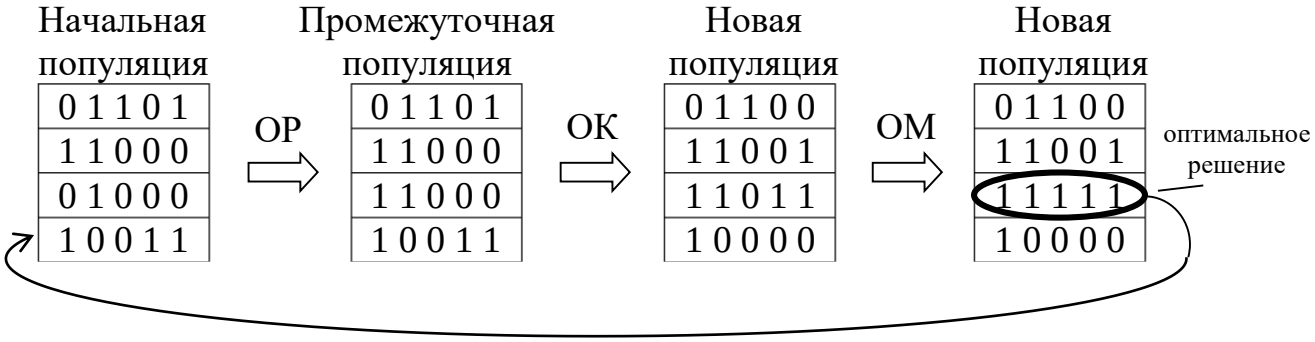

Рис.1.5 Процесс поиска решения «простым» ГА в примере Д.Голдберга

В результате $x = 31$, а ЦФ $f(x) = 961$, т.е. мы нашли оптимальное решение на отрезке [0,31].

1.3 Представление вещественных решений в двоичной форме для ГА

До этого в предыдущем примере мы рассматривали только целочисленные решения. Обобщим ГА на случай вещественных чисел. Рассмотрим следующий пример, в котором для функции

f(x)=(1,85 -x)*cos(3,5x – 0,5) (рис.1.6)

необходимо найти вещественное $x \in [-10,+10]$, которое максимизирует f, т.е. такое x_0, для которого $f(x_0) \ge f(x)$ для всех $x \in [-10,+10]$.

Нам надо построить ГА для решения этой задачи. Для представления вещественного решения (хромосомы) x будем использовать двоичный вектор, который применяется в классическом простом ГА. Его длина зависит от

требуемой точности решения, которую в данном случае положим 3 знака после запятой.

Если a=-10, c=10 – соответственно левая и правая границы области решения $x \in [a, c]$ и k=3 – заданная точность решения, то для достижения заданной точности отрезок должен быть разбит на $\frac{c-a}{10^{-k}} = (c-a)10^k$ равных частей. Т.е. в нашем случае на $(10+10)10^3 = 20000$ равных частей.

В качестве двоичного представления используем двоичный код номера отрезка. Этот код позволяет определить соответствующее ему вещественное число, если известны границы области решения. Отсюда следует, что двоичный вектор для кодирования вещественного решения должен иметь 15 бит, поскольку

$16384 = 2^{14} < 20000 \le 2^{15} = 32768$

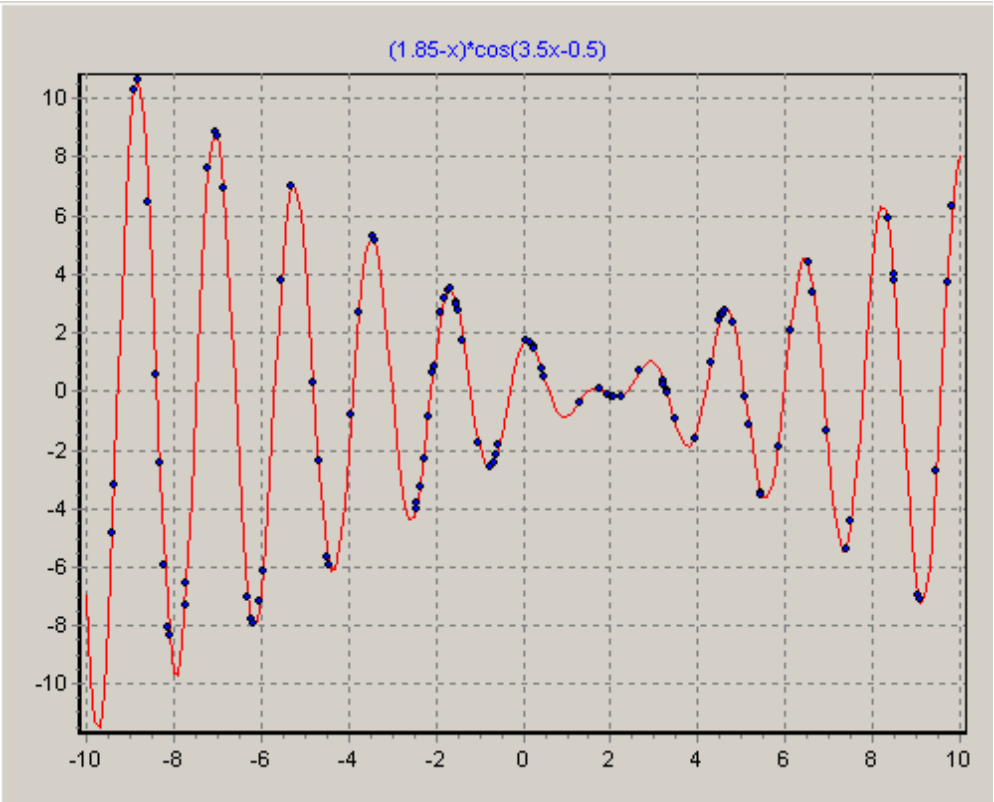

Рис.1.6

Отображение из двоичного представления $(b_{14}b_{20}...b_0)$ $(b_i \in \{0,1\})$ в вещественное число из отрезка [-10,+10] выполняется в два шага.

1) Перевод двоичного числа в десятичное:

$$(< b_{14}b_{13}...b_0 >)_2 = \left(\sum_{i=0}^{14} b_i 2^i\right)_{10} = X'$$

2) Вычисление соответствующего вещественного числа x:

$x = a + x' \cdot \frac{(c-a)}{2^n - 1} = -10 + x' \cdot \frac{20}{2^{15} - 1}$, где a=-10, c=10 – соответственно левая и правая границы области решения.

Естественно хромосомы

(000000000000000) и (111111111111111)

представляют границы отрезка –10 и +10 соответственно.

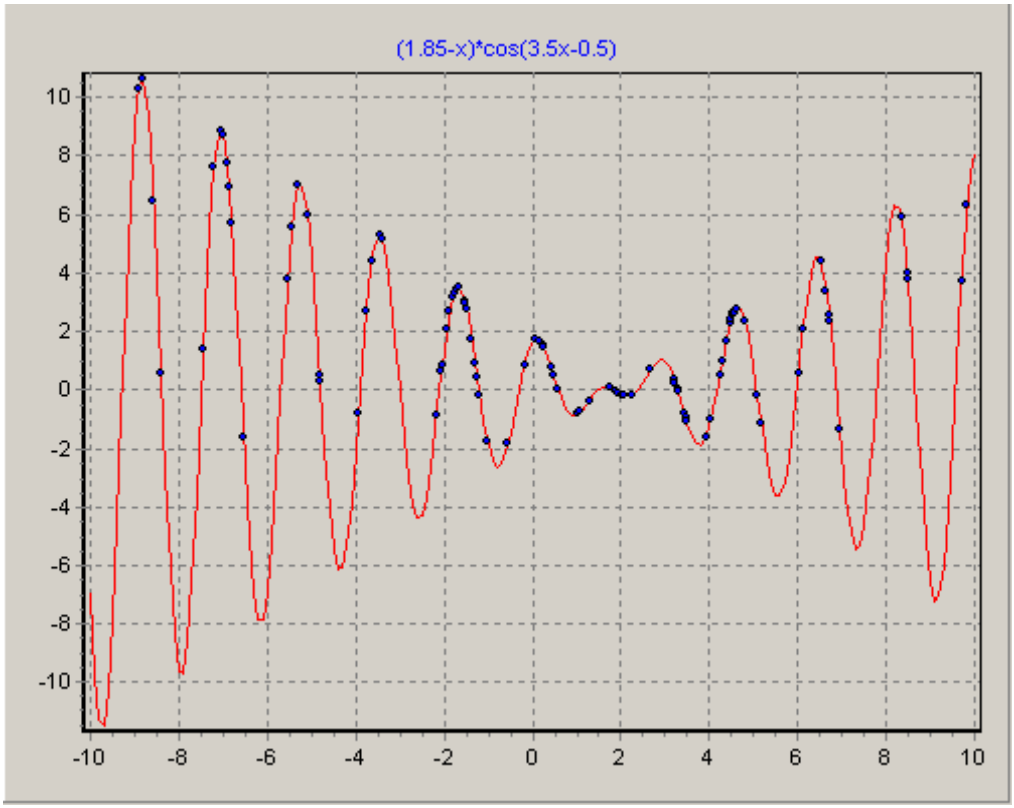

Рис.1.7

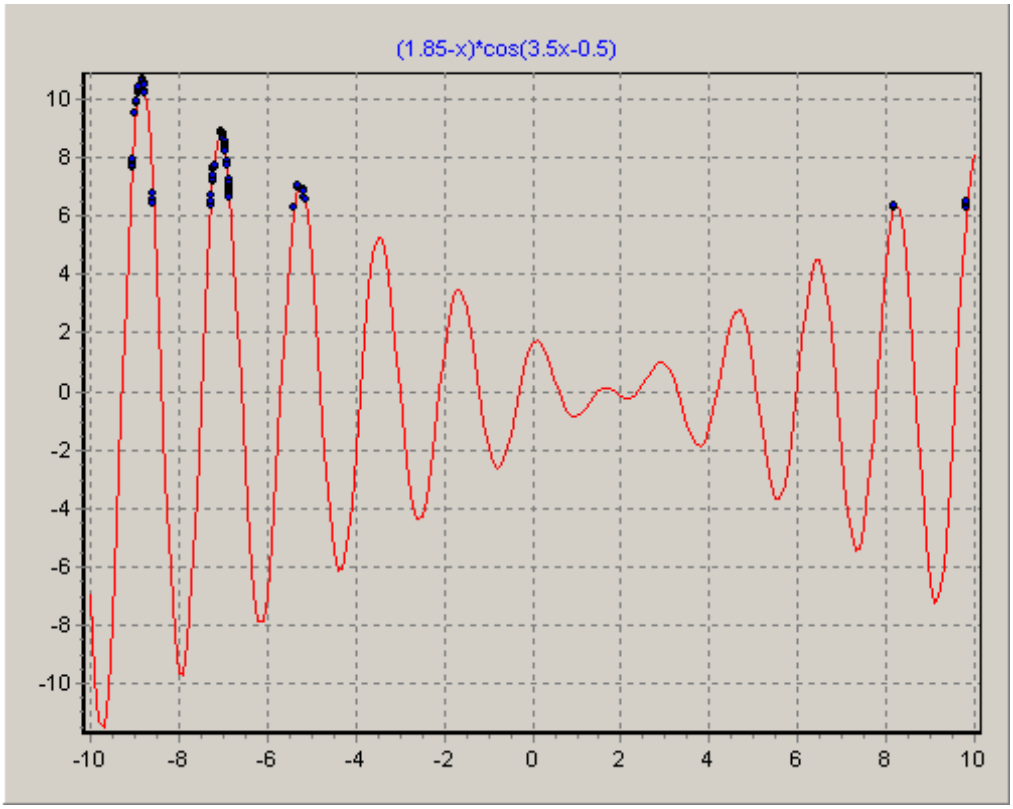

Рис.1.8

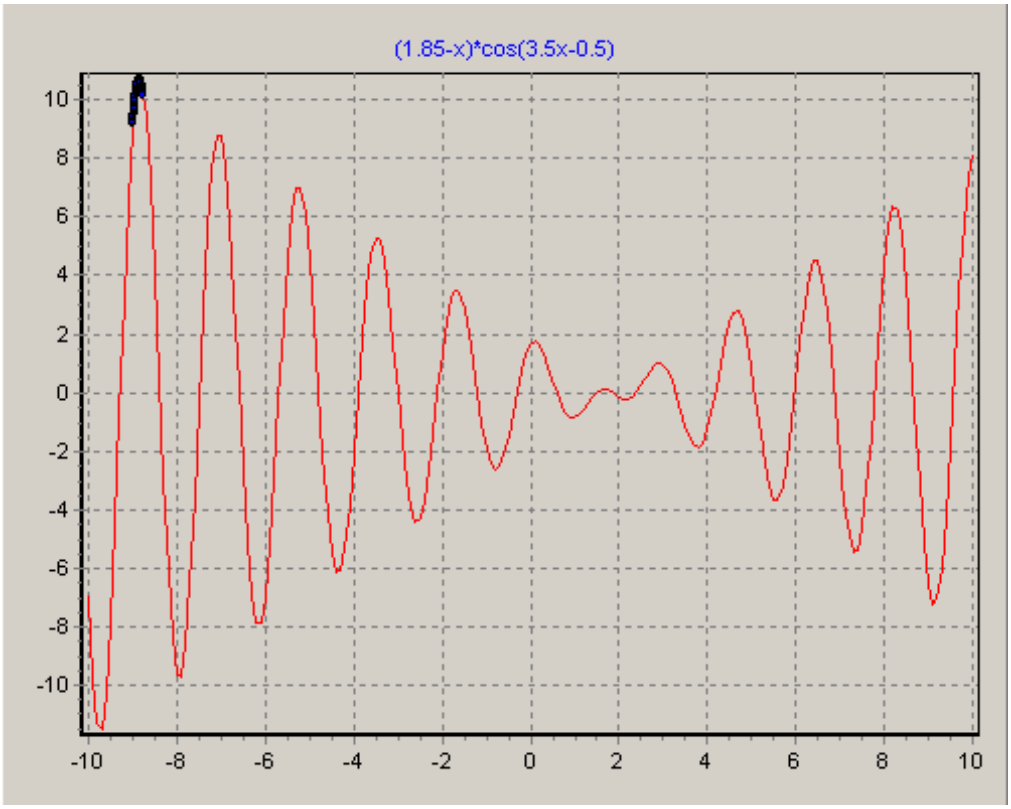

Рис.1.9

Очевидно, при данном двоичном представлении вещественных чисел можно использовать классический простой ГА. На рис.1.6–рис.1.9 представлено

расположение особей - потенциальных решений на различных этапах ГА в процессе поиска решения. На рисунке 1.6 показана начальная популяция потенциальных решений, которая равномерно покрывает область поиска решения. Далее явно видно, как постепенно с увеличением номера поколения особи "конденсируются" в окрестностях экстремумов и в конечном счете находится лучшее решение.

В заключение отметим, что ГА отличаются от других оптимизирующих и поисковых процедур следующими особенностями:

1) Работают не с параметрами, а с закодированным множеством параметров.
2) Осуществляет поиск из популяции точек, а не из единственной точки.
3) Использует целевую функцию непосредственно, а не ее непосредственное приращение.
4) Использует не детерминированные, а вероятностные правила поиска решений.
5) Каждая новая популяция состоит из жизнеспособных особей (хромосом).
6) Каждая новая популяция лучше (в смысле целевой функции) предыдущей.
7) В процессе эволюции последующая популяция зависит только от предыдущей.

Цель ГА двоякая:

1) Абстрактно и формально объяснить адаптацию процессов в естественных системах.
2) Спроектировать искусственные программные и технические системы на основе механизмов, использующихся в естественных системах.

1.4 Использование кода Грея в ГА

Рассмотренное двоичное представление вещественного числа, имеет существенный недостаток: расстояние между числами (на числовой оси) часто не соответствует расстоянию (по Хеммингу) между их двоичными представлениями. Поэтому желательно получить двоичное представление, где близкие расстояния между хромосомами (двоичными представлениями) соответствовали близким расстояниям в проблемной области (в данном случае

расстоянию на числовой оси). Это можно сделать, например, с помощью кода Грея.

Код Грея — непозиционный код с одним набором символов (0 и 1) для каждого разряда. Таким образом, в отличие от римской системы счисления число в коде Грея не является суммой цифр. Чтобы показать соответствие последовательности чисел коду Грея можно воспользоваться таблицей, но есть и наглядное правило построения этой последовательности. Младший разряд в последовательности чисел в коде Грея принимает значения 0 и 1, затем следующий старший разряд становится единичным и младший разряд принимает свои значения уже в обратном порядке (1, 0). Этим и объясняется название кода — отраженный. Соответственно, два младших разряда принимают значения 00, 01, 11, 10, а затем, при единичном следующем старшем разряде, те же значения в обратном порядке (10, 11, 01, 00). Ниже дана таблица, показывающая первые восемь чисел в двоичном коде и в коде Грея.

Приведем для примера код Грея, для 4-х битовых слов.

Таблица 1.4

Двоичный код	Код Грея
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

Заметим, что в коде Грея соседние двоичные слова отличаются на 1 бит (расстояние по Хеммингу = 1).

Рассмотрим алгоритмы преобразования двоичного числа $\bar{b} =< b_1,...,b_m >$ в код Грея $\bar{g} =< g_1,...,g_m >$ и наоборот.

```
Procedure Binary-to-Gray()
 {
 |      g1=b1
 |      for k=2 to m do
 |              gk=bk-1 xor bk
  }

Procedure Gray-to-Binary
  {
 |      value=g1
 |      b1=value
 |      for k=2 to m do
 |              begin
 |                  |       if gk=1 then value=NOTvalue
 |                  |       bk=value
  }
end
```

Рис.1.10

Здесь параметр m определяет разрядность двоичного числа.

Существует и другая матричная (эквивалентная) процедура преобразования в код Грея. Например, для m=4 матрица

$$A = \begin{bmatrix} 1000 \\ 1100 \\ 0110 \\ 0011 \end{bmatrix} \text{ и } A^{-1} = \begin{bmatrix} 1000 \\ 1100 \\ 1110 \\ 1111 \end{bmatrix}$$

позволяют выполнять следующие преобразования: $\bar{g} = A \cdot \bar{b}$ и $\bar{b} = A^{-1}\bar{g}$, где умножение матриц выполняются по mod 2. Отметим, что код Грея прежде всего оправдывается при использовании операторов мутации.

2. Теория схем ГА. Фундаментальная теорема ГА

Теоретические основы ГА представляют двоичное строковое представление решений (хромосом) и понятие схемы (schema) или шаблона. Понятие «схема» согласно Холанду [1] есть шаблон, описывающий подмножество строк, имеющих одинаковые значения в некоторых позициях. Для этого вводится новый троичный алфавит {0, 1, *}, где * означает неопределенное значение (0 или 1, то неизвестно что именно). Например, схема (0*0001) соответствует двум строкам {000001, 010001}, а (*0110*) описывает подмножество из 4-х строк {00100, 101100, 001101, 101101}. Очевидно, схема с m неопределенными позициями “*” представляет 2^m строк. Для строк длины n, существует 3^n схем (возможны 3 символа {0, 1, *} в каждой из n позиций).

Для геометрического представления возьмем куб в трехмерном пространстве (Рис.2.1). Обозначим вершины этого куба трехразрядными бинарными строками так, чтобы метки соседних вершин отличались ровно на один разряд, причем вершина с меткой "000" находилась бы в начале координат.

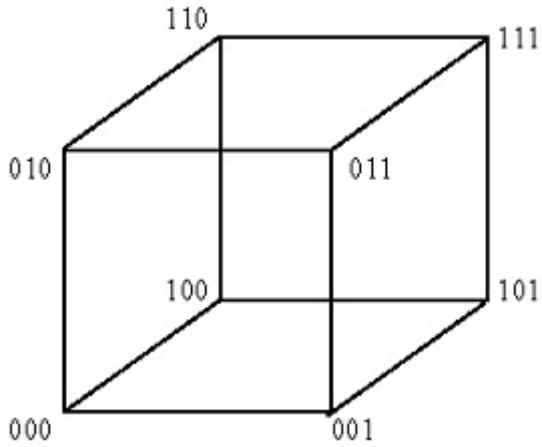

Рис.2.1 Куб в 3-мерном пространстве

Если взять схему вида "∗ ∗ 0", то она опишет левую грань куба, а схема "∗10" — верхнее ребро этой грани. Очевидно, что схема "∗ ∗ ∗" соответствует всему пространству.

Если взять двоичные строки длиной 4 разряда, то разбиение пространства схемами можно изобразить на примере четырехмерного куба с

поименованными вершинами (Рис.2.2). Здесь схеме "∗1 ∗ ∗" соответствует гиперплоскость, включающая задние грани внешнего и внутреннего куба, а схеме "∗ ∗ 10" — гиперплоскость с верхними ребрами левых граней обоих кубов. Таким образом, термины «гиперплоскость» и «схема» взаимозаменяемы.

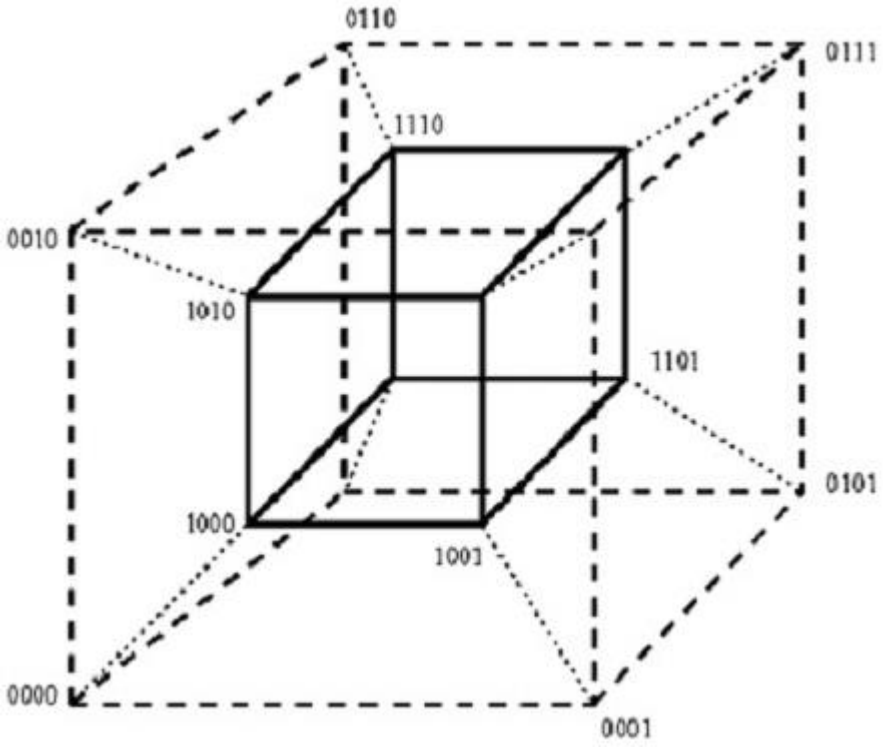

Рис.2.1 Куб в 4-мерном пространстве

В простом ГА основная идея заключается в объединении хромосом со значениями ЦФ выше среднего. Например, пусть 1 в хромосоме соответствует наличию признака, способствующего выживанию (значение ЦФ больше среднего). Допустим, что имеются подстроки вида 11*** и **111. Тогда, применяя к ним ОК можно получить хромосому 11111 с признаками, способствующим наилучшим значениям фитнесс функции. Рассмотрим также полезное понятие строительного блока. Например, в шаблоне ****1 строительным блоком будет элемент 1. В шаблоне 10*** строительным блоком будет составной элемент 10. Очевидно, вид строительных блоков должен выбираться из знаний о решаемой задаче и отражать полезные свойства, чтобы далее из строительных блоков как из “кирпичиков” собрать “здание”, т.е.

решение с лучшей ЦФ. Желательно, чтобы в ГА выражались условия, которые разрешают разрыв строительных блоков только в крайних случаях, указанных пользователем.

Не все схемы являются одинаковыми. Некоторые более специфичны, чем другие. Для количественной оценки вводятся 2 характеристики:

1) порядок схемы O(H), который определяется числом фиксированных позиций (не равных *). Например, для H_1 = 0*1*** ее порядок $O(H_1) = 2$
2) Определенная длина L(H) – расстояние между первой и последней определенной (не равной *) позицией. Например для шаблона H_2 = 0111*1**

   $L(H_2) = 6 – 1 = 5$. А для H_3 = 0**** $O(H_3) = 1 – 1 = 0$.

Обозначим через m(H,t) – число строк, содержащихся в популяции A(t) (t – шаг итерации или время), которые отображаются (покрываются) схемой H.

Пусть *f(H)* означает среднее значение ЦФ хромосом, покрываемых данной схемой H, а

$\overline{f}(x) = \frac{\sum_{j=1}^{N} f(x_j)}{N}$ – среднее значение ЦФ для всей популяции (N – размер популяции).

Рассмотрим, как ведет себя m(H,t) при выполнении основных операторов ГА. Нижеприведенные результаты составляют фундаментальную теорему ГА – теорему схем[1].

2.1 Влияние репродукции

Напомним, что в процессе репродукции хромосомы копируются в промежуточную популяции согласно их значениям ЦФ –хромосома x_i со значением $f(x_i)$ выбирается с вероятностью $P(x_i) = \frac{f(x_i)}{\sum f(x_j)}$. После селекции мы ожидаем на следующем шаге получить m(H,t+1) стрингов, отображаемой схемой H.

Известно, что

$$m(H,t+1) = m(H,t)*N*f(H)/\sum f(x_j) \qquad (2.1)$$

Это обусловлено тем, что:

(1) в среднем вероятность выбора строк, представляемых схемой H, определяется $f(H)/\sum f(x_j)$,

(2) число строк, представляемых H , равно m(H,t),

3) число строк в популяции равно N.

Мы можем переписать эту формулу с учетом обозначения

$$\overline{f}(x) = \frac{\sum_{j=1}^{N} f(x_j)}{N}$$

и получим следующее выражение для числа особей, покрываемых схемой в промежуточной популяции

$$m(H, t+1) = m(H, t) * f(H)/ \overline{f(x)}. \qquad (2.2)$$

Другими словами схемы “растут” как отношение среднего значения ЦФ схемы к среднему значению ЦФ популяции. Схема со значением ЦФ выше средней в популяции будет получать больше возможностей для копирования и наоборот.

Правило репродукции Холланда гласит “схема со значением ЦФ выше среднего живёт и размножается, а со зачением ЦФ ниже среднего умирает”. Предположим, что схема H остаётся со значением выше среднего ЦФ на величину $c*\bar{f}$, где c – константа. Тогда последнее выражение (2.2) можно модифицировать:

$$m(H, t+1) = m(H, t) * (\bar{f} + c * \bar{f}) / \bar{f} = (1 + c)\, m(H, t).$$

Начиная с t = 0 и предполагая, что c – величина постоянная, получаем следующее выражение числа особей промежуточной популяции, покрываемых схемой,

$$m(H,t) = m(H,0)* (1+c)^t \ .$$

Очевидно, что при c>0 схема “размножается”, а при c<0 схема умирает. Далее рассмотрим влияние оператора кроссинговера на число особей в популяции, покрываемых схемой.

2.2 Влияние кроссинговера

Рассмотрим конкретную строку длины n = 7 A = 011| 1000 и две схемы, представляющие эта строка:

H1 = *1*| ***0, H2 = ***| 10**.

Здесь символ ‘|’ – , как обычно, обозначает точку кроссинговера (k=4).

Очевидно, что схема H1 после ОК с точкой k=4, скорее всего, будет уничтожена потому, что ‘1’ в позиции 2 и ‘0’ в позиции 7 попадут в разные новые строки после кроссинговера. С другой стороны, ясно, что схема H2 будет выживать, так как “10” в позициях 4,5 будут содержаться вместе в одном новом строке. Хотя мы взяли точку скрещивания ОК случайно, ясно, что схема H1 менее приспособлена к выживанию, чем схема H2. Если точка скрещивания ОК выбирается случайно среди n-1 = 7-1 = 6 возможных позиций, то ясно, что схема H1 разрушается с вероятностью

P(d) = L(H1) / (L-1) = 5/6.

Очевидно, что эта же схема выживает с вероятностью

P(S) = 1-P(d) = 1/6.

Аналогично, схема H2 имеет L(H2) = 1 и вероятность её уничтожения P(d) = 1/6, а вероятность её выживания схемы после применения ОК P(S) = 5/6. Очевидно, что нижняя граница вероятности выживания схемы после применения ОК может быть вычислена для любой схемы. Так как схема выживает, когда точка ОК попадает вне “определенной длины”, вероятность выживания для простого ОК определяется по формуле

$P_s(OK) = 1 – L(H)/(L-1)$.

Если ОК выполняется посредством случайного выбора, например, с вероятностью $P_{ОК}$, то вероятность выживания схемы определяется так:

$P(S) \geq 1 – P_{ОК}*L(P)/(L-1)$.

Очевидно, что это выражение уменьшается при $P_{OK} \to 1$. Теперь мы можем асимптотически оценить эффект операторов репродукции и кроссинговера.

При независимости выполнения ОР и ОК можно получить следующее выражение:

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}[1 - P_{OK}\frac{L(H)}{L-1}].$$

Таким образом, число схем H в новой популяции зависит от двух факторов:

1) значение ЦФ схемы выше или ниже ЦФ популяции;

2) схема имеет “длинную” или “короткую” L(H) (определенную длину).

Видно, что схемы со значением ЦФ выше средней и короткой длиной L(H) имеют возможность экспоненциального роста в новой популяции. Далее рассмотрим влияние оператора мутации на число особей в популяции, покрываемых схемой.

2.3 Влияние мутации

Напомним, что оператор мутации ОМ есть случайное изменение элемента в строке с вероятностью P_{OM}. Очевидно, что для того, чтобы схема H выжила, все определенные позиции должны выжить. Поскольку единственный ген выживает с вероятностью $(1\text{-}P_{OM})$, то данная схема выживает, когда каждая из O(H) фиксированных позиций схемы выживает. Умножая вероятность выживания $(1\text{-}P_{OM})$ O(H) раз, получим, что вероятность выживания при ОМ равна $(1-p_{OM})^{O(H)}$. Для малых величин $P_{OM} << 1$ это выражение может быть аппроксимировано

$P_S(OM) = 1 – O(H)\cdot P_{OM}$.

Поскольку вероятность выживания равна (1- вероятности разрушения при ОК и ОМ), то схема H дает ожидаемое число особей в следующей популяции после выполнения всех генетических операторов ОР, ОК и ОМ согласно следующей формуле

$$m(H,t+1) > m(H,t) \cdot \frac{f(H)}{\bar{f}}[1 - P_{OK} \frac{L(H)}{1-L} - O(H) \cdot P_{OM}] \qquad (2.3)$$

2.4 Фундаментальная теорема ГА.

Таким образом, на основе выше изложенного формулируем следующую теорему:

Для данной схемы H ожидаемое число особей в следующей популяции после выполнения всех генетических операторов ОР, ОК и ОМ определяется согласно неравенству (2.3).

Этот важный результат известен как фундаментальная теорема ГА или теорема схем. Теорема схем показывает, что строительные блоки растут по экспоненте, в то время схемы с приспособленностью ниже средней распадаются с той же скоростью. Голдберг в своих исследованиях теоремы схем выдвигает гипотезу строительных блоков, которая состоит в том, что «строительные блоки объединяются, чтобы сформировать лучшие строки» [2]. То есть, рекомбинация и экспоненциальный рост строительных блоков ведет к формированию лучших строительных блоков.

Вернемся к примеру Голдберга определения $\max f(x) = x^2$. В дополнение к имеющимся таблицам 1.1-1.3 пусть имеется 3 конкретные схемы H1=1*****, H2=*10** и H3=1***0, описанные в следующей таблице.

Таблица 2.1

Схема	Перед репродукцией	Представители стрингов	Средняя ЦФ схемы f(H)
H1	1****	2,4	469
H2	*10**	2,3	320
H3	1***0	2	576

Таблица 2.2

Схема	После репродукции			После всех операторов		
	Ожидаемое число строк	Действительное число строк	Представители строк	Ожидаемое число строк	Действительное число строк	Представители строк
H1	3,20	3	2, 3, 4	3,20	3	2, 3, 4
H2	2,18	2	2, 3	1,64	2	2, 3
H3	1,97	2	2, 3	0,0	1	4

Рассмотрим сначала схему H1. При репродукции строки копируются с вероятностью, определенной согласно величине их ЦФ. Из табл. 2.1 видно, что строки 2,4 покрываются схемой H1. После ОР мы видим табл. 2.2 (строка H1), что получены три копии и строка 3 также вошла в популяцию. Проверим, соответствует ли это число фундаментальной теореме схем. Согласно теории мы ожидаем $m*f(H)/\bar{f}$ копий . Вычисляя среднее значение ЦФ f(H1) получаем $\bar{f}$ (H1)=(576+361)/2=468,5. Разделив это число на среднее значение ЦФ популяции $\bar{f}$=293 и умножив на число строк, покрываемых H1 на шаге t m(H1,t)=2, получаем ожидаемое число строк, покрываемых H1 на шаге t+1, т.е.

$$m(H1,t+1)=2\cdot 468,5 / 293 = 3,20.$$

Сравнивая это число с реальным числом копий 3, видим, что округление рассчитанного значения копий 3,2 дает реальное их число 3. Дальнейший анализ показывает, что в данном случае ОК не оказывает влияние на число строк, покрываемых схемой, так как определенная длина L(H1)=0 предотвращает разрыв схемы. Далее при мутации с вероятностью P_{OM}=0,001 мы должны ожидать уменьшение числа стрингов на m*O(H1)*P(OM)=3*1*0,001=0,003, что практически не оказывает влияния на

окончательный результат. В итоге для схемы H1 получаем ожидаемое увеличение числа особей до 3, что соответствует формуле 2.2.

Рассмотрим теперь схемы H2 и H3 с двумя фиксированными позициями. Схема H2 имеет также два представителя в начальной популяции (2,3) и имеет две копии в следующей промежуточной популяции. Вычисляем m(H2)=2·320/293=2,18. Здесь f(H2)=320 – среднее значение ЦФ схемы, а $\bar{f}$=293 – среднее значение ЦФ популяции. Для H3 получаем только одно стринговое представление и m(H3)=1·576/293=1,97. Здесь 576 – среднее значение ЦФ схемы.

Заметим, что для конкретной схемы H2 две копии – это хороший результат и он может случиться только 1 раз из 4-х возможных случаев (L-1=5-1)=4. Схема H2=*|1|0|*|* - имеет реальную возможность дать новую копию. В результате, схема H2 выживает с высокой вероятностью. Действительно, число стрингов, покрываемых H2, равно m(H2,t+1)=2,18·0,75=1,64≈2.

Для схемы H3 при высокой определенной длине L(H3)=4 оператор ОК обычно разрушает эту схему. Поэтому ожидаемое число m(H3,t+1)=0 в соответствии с формулой (2.3).

3. Модификации и обобщения генетических алгоритмов

Мы изучили в предыдущих разделах, в основном, классическую реализацию простого ГА. Далее рассмотрим различные модификации и обобщения для каждого функционального блока основной блок-схемы ГА (Рис.1.1).

3.1 Создание исходной популяции

Возможны 3 способа создания исходной популяции (начального множества решений).

Стратегия " одеяла" – формирование полной популяции, содержащей все возможные решения. Например, для n-разрядной хромосомы мы имеем всего 2^n вариантов решений, которые составляют полную популяцию.

Стратегия " дробовика*"* - *г*енерация достаточно большого случайного подмножества решений.

Стратегия фокусировки - *г*енерация множества решений, включающих разновидности одного решения.

Первый подход (стратегия «одеяла») практически не реализуем даже для задач средней размерности, вследствие больших вычислительных затрат. Заметим, что при этом подходе начальная популяция не может развиваться, т.к. в ней уже содержатся *все* возможные решения.

Третий способ используется тогда, когда есть предположение, что некоторое решение является вариацией известного значения. В этом случае время поиска оптимального решения существенно сокращается, т.к. алгоритм начинает работу в окрестности оптимума.

Чаще всего применяют второй способ. В этом случае в результате эволюции есть возможность перейти в другие подобласти поиска и каждая из них имеет сравнительно небольшое пространство поиска.

В целом эффективность ГА, качество решения и дальнейшая эволюция в значительной степени определяются структурой и качеством начальной популяции.

На практике часто применяют комбинацию второго и третьего способов. Сначала определяются особи с высокими значениями целевой функции, а затем случайно формируются начальные решения в этих подобластях.

3.2 Отбор родителей (селекция)

Оператор отбора S порождает промежуточную популяцию $\widetilde{P}^t$ из текущей популяции P^t путем отбора и генерации новых копий особей.

$$P^t \xrightarrow{S} \widetilde{P}^t$$

При отборе конкурентно-способных особей используют целевую (fitness) функцию, как единственно доступный источник информации о качестве решения. Но различные методы отбора родителей по-разному используют эту информацию. Далее мы рассмотрим наиболее распространенные методы отбора родителей.

3.2.1 Пропорциональный отбор (метод " рулетки")

Этот вид отбора чаще всего используется на практике. При этом особи (решения) отображаются в отрезки линии (или сектора рулетки) таким образом, что их размер пропорционален значению целевой функции для данной особи. Это показано в следующем примере, представленном в табл.3.1

Таблица 3.1

Номер особи	1	2	3	4	5	6	7	8	9	10	11
Значение ЦФ	2,0	1,8	1,6	1,4	1,2	1,0	0,8	0,7	0,3	0,2	0,0
Вероятность выбора	0,18	0,16	0,15	0,13	0,11	0,09	0,07	0,06	0,03	0,02	0,00

На прямой строим отрезки, длины которых пропорциональны вероятностям выбора особей, как это показано на Рис.3.1.

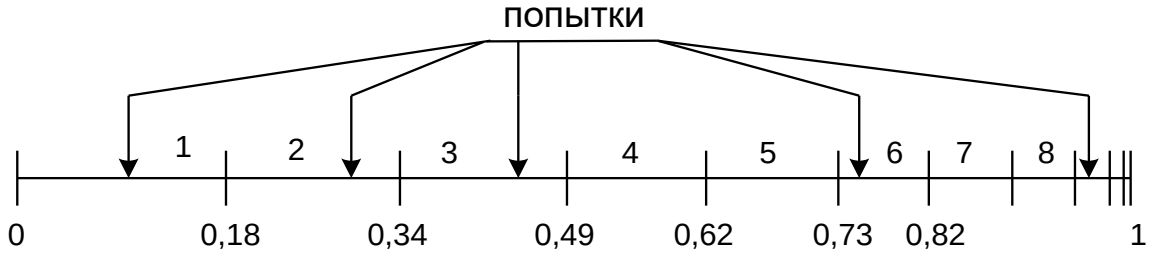

Рис.3.1

Далее случайно генерируются числа в интервале от 0.0 до 1.0 и в промежуточную популяцию выбираются те особи, в "чей отрезок" попадают эти случайные числа. Таким образом, каждая попытка представляет собой случайное число в интервале от 0 до 1, в результате чего выбирается та особь, в чей отрезок попадает это число. В данном примере в результате пяти попыток были выбраны в промежуточную популяцию особи 1, 2, 3, 6 и 9.

Данный метод, не смотря на частое применение, имеет следующие недостатки:

– зависимость от положительных значений целевой функции (целевая функция должна для всех особей принимать положительное значение);

– простое добавление очень большой константы к целевой функции, может свести способ отбора к равновероятному выбору. Так, добавляя к значению целевой функции в таблице 3.1 значение 1000, получаем равновероятностный выбор (Табл.3.2).

Таблица 3.2

Номер особи	1	2	3	4	5	6	7	8	9	10	11
Значение ЦФ	1002	1001,8	1001,6	1001,4	1001,2	1001	1000,8	1000,7	1000,3	1000,2	1000
Вероятность выбора	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09

Для приведенного примера можно привести следующую гистограмму вероятностей выбора особей, представленную на Рис.3.2. Отметим, что здесь высота "ступенек " различна, в отличие от следующего метода выбора, для которого гистограмма представлена на Рис.3.3.

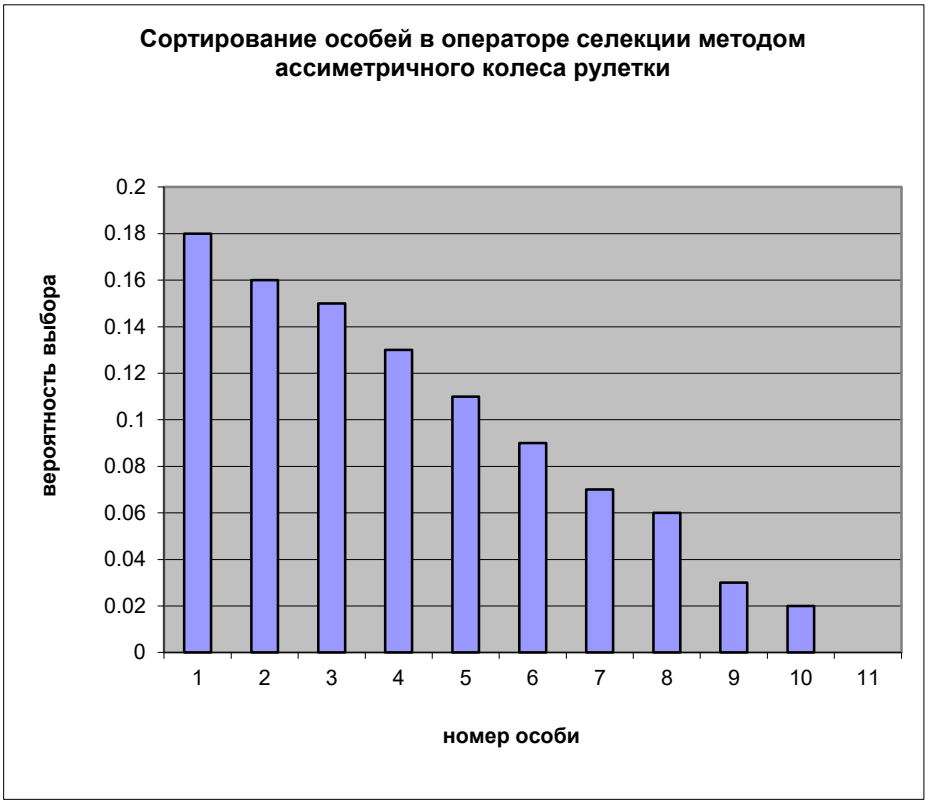

Рис.3.2

Для устранения первых двух недостатков используют масштабирование оценок значений целевой функции:

а) сдвиг (в случае поиска минимума функции и оптимального маршрута)

$$F1(x_i) = \max_i F(x_i) - F(x_i)$$

б) линейное (для выделения лучших особей)

$$F1(x_i) = a + bF(x_i),$$

$$a = \frac{\frac{1}{K}\sum_{i=1}^{K} F(x_i)}{\frac{1}{K}\sum_{i=1}^{K} F(x_i) - \min_i F(x_i)}, \; b = -\frac{\left(\frac{1}{K}\sum_{i=1}^{K} F(x_i)\right)\min_i F(x_i)}{\frac{1}{K}\sum_{i=1}^{K} F(x_i) - \min_i F(x_i)},$$

причем b a, удовлетворяют трем условиям

$$\frac{1}{K}\sum_{i=1}^{K} Fl(x_i) = \frac{1}{K}\sum_{i=1}^{K} F(x_i);$$

$$\frac{\max\limits_i Fl(x_i)}{\frac{1}{K}\sum_{i=1}^{K} Fl(x_i)} = 2;$$

$$\min_i Fl(x_i) = 0,$$

где K – мощность популяции;

в) сигма-отсечение (для выделения лучших особей)

$$Fl(x_i) = \max\left(0, F(x_i) + (m - c\sigma)\right),$$

где m, σ – математическое ожидание и среднеквадратичное отклонение по популяции, $c \in [1,5]$ – натуральная константа;

г) степенное (для выделения лучших особей)

$$Fl(x_i) = F^k(x_i),$$

где $k > 1$ – константа (обычно $k = 2$ или 3);

д) Больцмана (имитация отжига) (для выделения лучших особей)

$$Fl(x_i) = \exp\left(\frac{F(x_i)}{T(n)}\right),$$

где $T(n)$ – температура на итерации n, например,

$$T(n) = \frac{1}{\sqrt{n}} \quad \text{или}$$

$$T(n) = \beta T(n-1),\ 0 < \beta < 1,\ g(0) = T_0,\ T_0 > 0.$$

Затем особи упорядочиваются по фитнесс-функции и определяется вероятность выбора каждой i -й хромосомы в виде

$$P(x_i) = \frac{Fl(x_i)}{\sum_{s=1}^{K} Fl(x_s)},\ i \in \overline{1,K}.$$

3.2.2 Ранжирование

При этом методе особи популяции сортируются (упорядочиваются) согласно значениям целевой функции. Отметим, что здесь вероятность отбора для каждой особи зависит только от ее позиции (номера) в этом упорядоченном множестве особей, а не от самого значения целевой функции. Этот метод отбора более устойчив, чем предыдущий. В основном, применяют линейное ранжирование, где вероятность отбора определяют в соответствии с выражением.

$$P_i(a_i) = \frac{1}{N}(a - (a-b)\frac{i-1}{N-1}),$$

где $1 \le a \le 2$ выбирается случайным образом, $b = 2 - a$, N – мощность популяции, i – номер особи.

Для приведенного выше примера значения вероятностей выбора особей приведены в таблице 3.3, а соответствующая гистограмма представлена на рис.3.3. Отметим, что здесь высота "ступенек" одинакова, в отличие от предыдущего метода выбора. Небольшое отличие может присутствовать вследствие округления результата.

Таблица 3.3

Номер особи	1	2	3	4	5	6	7	8	9	10	11
Значение ЦФ	2	1,8	1,6	1,4	1,2	1	0,8	0,7	0,3	0,2	0
Вероятность выбора	0,155	0,142	0,129	0,116	0,104	0,091	0,078	0,065	0,053	0,04	0,027

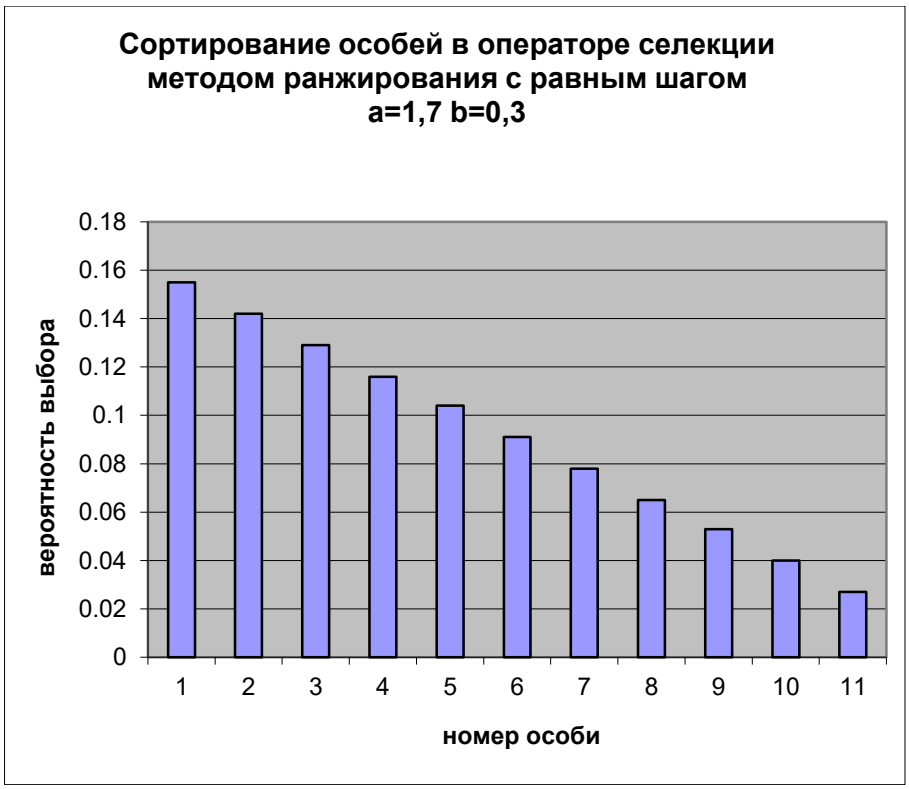

Рис.3.3

Иногда применяют нелинейное ранжирование. При этом вероятность отбора также определяется номером позиции в упорядоченном множестве особей, но вид функции может быть сложнее.

3.2.3 Равномерное ранжирование (случайный выбор).

Здесь вероятность выбора особи определяется выражением:

$$P_S(a_i) = \begin{cases} \dfrac{1}{\mu} & при \quad 1 \le i \le \mu \\ 0 & при \quad \mu < i \le N \end{cases}$$

где $\mu \le N$ – параметр метода.

Для методов ранжирования и равномерного ранжирования также на прямой строим отрезки, длины которых пропорциональны вероятностям выбора особей, как это показано на Рис.3.1.

Далее случайно генерируются числа в интервале от 0.0 до 1.0 и в промежуточную популяцию выбираются те особи, в "чей отрезок" попадают

эти случайные числа. Таким образом, каждая попытка представляет собой случайное число в интервале от 0 до 1, в результате чего выбирается та особь, в чей отрезок попадает это число.

3.2.4 Локальный отбор.

Производится среди особей, которые находятся в некоторой ограниченной среде, где определено отношение соседства. Ранее, фактически, в качестве соседей каждой особи рассматривалась вся популяция. При этом соседи рассматриваются как множество партнеров для выполнения операции скрещивания.

Соседство можно определить по-разному. Далее рассмотрим типичные отношения соседства, используемые при локальном отборе (и не только в нем).

1) Линейное соседство:

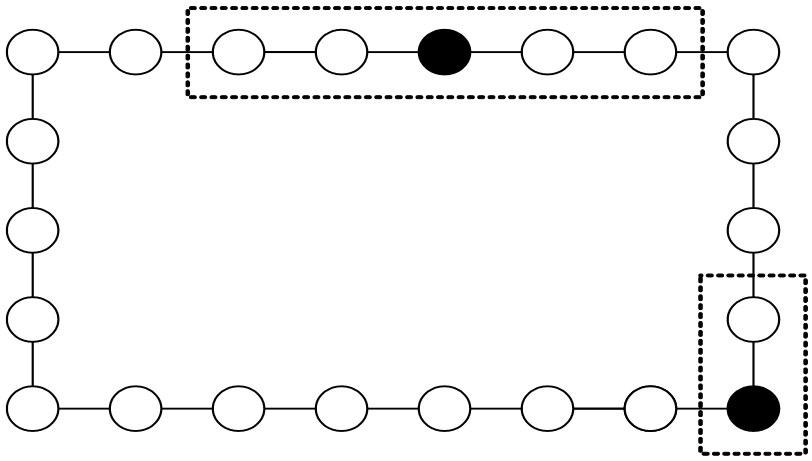

Рис.3.4

На рис.3.4 представлен пример линейного соседства. На практике рассматривают полную окрестность (на рис.3.4 вверху показана полная окрестность выделенного элемента с расстоянием d=2) и полуокрестность (в правом нижнем углу рисунка показана полуокрестность с расстоянием d=1).

2) Двумерное – четырехсвязное соседство.

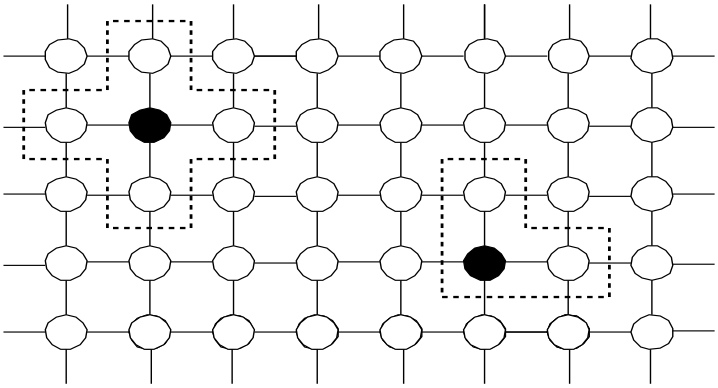

Рис.3.5

На рис.3.5 представлен пример соседства с 4-связностью (в левом верхнем углу показан полный "крест" выделенного элемента с расстоянием d=1, справа "полукрест" также с расстоянием d=1).

3) Двумерное – восьмисвязное соседство.

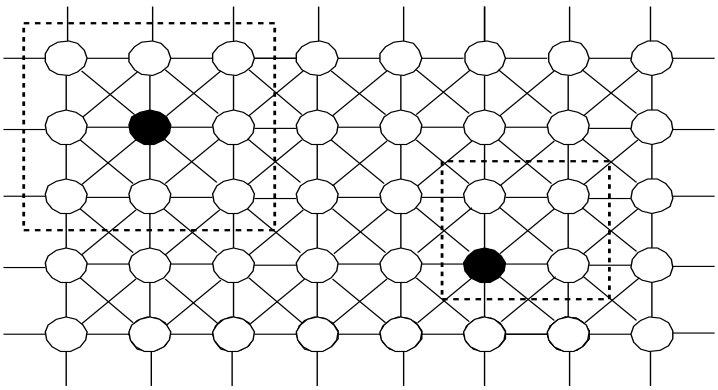

Рис.3.6

На рис.3.6 представлен пример соседства с 8-связностью (в левом верхнем углу показан полная "звезда" выделенного элемента с расстоянием d=1, а справа "полузвезда"). Для определения отношения можно также использовать гексагональную решетку с 6-связностью и даже 3-мерные структуры.

При отборе родителей на первом шаге производится отбор особей случайным образом, или одним из ранее рассмотренных способов. Далее, для каждой отобранной особи определяется множество локальных соседей и среди них выбирается партнер для выполнения операции скрещивания.

При наличии отношения соседства, между особями возникает «изоляция расстояния». Очевидно, что чем меньше соседство, тем больше «изоляция расстояния». Это ограничивает распространение новых решений в популяции.

Однако, из-за перекрытия соседних областей, распространение новых вариантов решений все же возможно. Мощность множества соседей определяет скорость распространения информации между особями популяции, способствуя либо быстрому распространению новых решений, либо сохранению имеющегося генофонда в популяции.

Часто при решении задачи требуется высокая изменчивость, которая поможет избежать преждевременной сходимости в районе локального оптимума. Обычно локальный отбор в малом окружении дает лучшие результаты, чем в большем окружении.

В малых и средних популяциях (N<100) для локального отбора рекомендуется двумерная структура типа полузвезда с расстоянием $d = 1.$

При большом размере популяции ($N > 100$) лучше использовать большие расстояния d >1 и 2-мерные структуры с соседством типа звезда.

3.2.5. Отбор на основе усечения.

Предыдущие методы отбора в какой-то степени заимствованы у (или имеют аналогию с) эволюции естественных популяций. Этот метод не имеет аналогов в естественной эволюции и обычно используется для больших популяций ($N > 100$). При этом сначала отбираемые особи упорядочиваются согласно их значениям целевой функции. Затем, в качестве родителей выбираются только лучшие особи. Далее, с равной вероятностью, среди них случайно выбирают пары, которые производят потомков.

При этом методе используется параметр – порог отсечения T (иногда используется термин интенсивность отбора), показывающий долю (часть популяции), которая отбирается в качестве родителей. Обычно $10\% \le T \le 50\%$

3.2.6. Турнирный отбор.

В этом случае из популяции случайно отбираются m особей, и лучшая из них выбирается в качестве родителя. Далее этот процесс повторяется, пока не сформируется промежуточная популяция, в которой случайно формируются пары родителей. Параметром этой процедуры является размер тура m, который принимает значения из диапазона $2 \le m < N$.

3.2.7. Методы выбора пар для скрещивания.

До сих пор мы, в основном, рассматривали методы отбора родителей в промежуточную популяцию. Далее из этой популяции необходимо выбрать пары особей для выполнения операции скрещивания. Основными методами выбора пар особей являются следующие.

Случайный выбор (панмиксия) родительской пары, при котором оба родителя случайным образом выбираются из всей популяции. Если не используется имитация отжига, то этот кроссинговер над выбранными родителями происходит обычно с вероятностью 0.5. Для имитации отжига кроссинговер происходит с вероятностью

$$p_c = p_0 \exp(-1/T(n)),\ T(n) = \beta T(n-1),\ 0 < \beta < 1, T(0) = T_0, T_0 > 0,$$

где p_0 – начальная вероятность кроссинговера, n – номер итерации.

Следует отметить, что при этом любая особь может входить в несколько пар. Этот метод является универсальным для решения различных задач, но достаточно критичен к численности популяции, поскольку его эффективность снижается с ростом мощности популяции N.

Селективный выбор. Здесь родителями могут стать только те особи, значение целевой функции которых не меньше среднего значения по популяции при равной вероятности этих кандидатов составить брачную пару. Такой подход обеспечивает более быструю сходимость алгоритма, но неприемлем для мультимодальных задач (имеющих несколько экстремумов), для которых этот метод, как правило, быстро сходится к одному из решений. Это может привести к преждевременной сходимости к локальному экстремуму.

Если не используется имитация отжига, то этот кроссинговер над выбранными родителями происходит обычно с вероятностью 0.5. Для имитации отжига кроссинговер происходит с вероятностью

$$p_c = p_0 \exp(-1/T(n)),\ T(n) = \beta T(n-1),\ 0 < \beta < 1, T(0) = T_0, T_0 > 0,$$

где p_0 – начальная вероятность кроссинговера, n – номер итерации.

Часто используется также два следующих подхода: *инбридинг* и *аутбридинг*. В них формирование пары происходит на основе близкого или дальнего родства соответственно. Под родством обычно понимается расстояние между особями популяции в пространстве параметров. В связи с этим различают генотипный и фенотипный инбридинг и аутбридинг.

Инбридинг – здесь первый член пары выбирается случайно, а вторым является максимально близкая к нему особь.

Аутбридинг формирует пары из максимально далеких особей.

Комбинация этих походов используется при решении многоэкстремальных задач.

3.3 Операторы рекомбинации (скрещивания, кроссинговера)

Различают операторы двоичной и вещественной рекомбинации (скрещивания). Рассмотрим сначала более привычную (и уже знакомую), применяемую в простом ГА двоичную рекомбинацию.

3.3.1 Двоичная рекомбинация

1) Одноточечный кроссинговер

Мы уже рассматривали его в разделе 1.1. Здесь случайно выбирается точка скрещивания с вероятностью $P \approx 0.5$ и производится обмен фрагментами хромосом после точки скрещивания скрещивание. Пример показан на следующем Рис.3.7.

а:	0	1	1	1	0	1	0	1	0
в:	1	0	0	1	1	0	1	0	1
						⇩			
а’:	0	1	1	1	0	0	1	0	1
в’:	1	0	0	1	1	1	0	1	0

Рис.3.7

2) Многоточечный кроссинговер.

В этом случайно выбираются m позиций $k_i \in [1,2,\ldots,n\text{-}1]$ и их номера сортируется по порядку. Затем родители обмениваются фрагментами, заключенными между соседними выбранными позициями и производят, таким образом, двух потомков. То есть обмен здесь идет секциями. При этом секции между первой позицией и первой точкой скрещивания не обмениваются, а далее обмен идет через одну секцию, как это показано на Рис.3.8.

а:	0	1	1	1	0	0	1	1	0	1	0
в:	1	0	1	0	1	1	0	0	1	0	1
а’:	0	1	1	0	1	1	1	1	0	1	1
в’:	1	0	1	1	0	0	0	0	1	0	0

Рис.3.8.

3) Однородный кроссинговер.

Этот вид скрещивания радикально отличается от предыдущих видов. Здесь каждый ген потомка создается путем копирования соответствующего гена из первого или второго родителя, то есть каждая позиция потенциально является точкой кроссинговера.

Для этого случайным образом генерируется двоичная маска кроссинговера той же длины (с тем числом бит), что у хромосом родителей. Создать маску можно следующим образом: введем некоторую величину $0 < p_0 < 1$, и если случайное число больше p_0, то на n-ю позицию маски ставим 1, иначе – 0. Таким образом, гены маски представляют собой случайные двоичные числа (0 или 1). Четность бита маски показывает родителя, из которого копируется ген потомка. Для определенности допустим, что 1 соответствует первому родителю, а 0 – второму. На Рис.3.9 показана схема выполнения этого типа кроссинговера на конкретном примере. Каждый бит потомка копируется из 1-го или 2-го

родителя в соответствии со значением этого бита маски. Таким образом, потомок содержит смесь генов из каждого родителя.

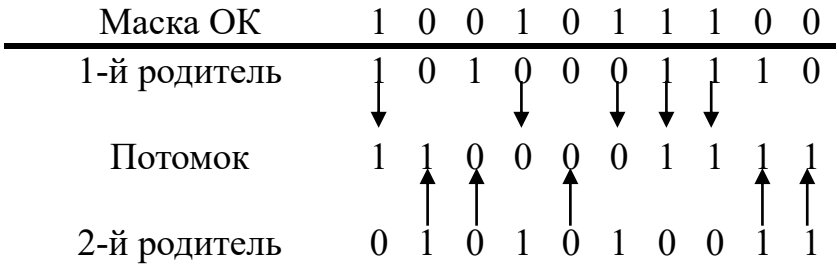

Рис.3.9

Однородный кроссинговер очень похож на многоточечный, но строка случайных битовых значений в нем длиннее. Это гарантирует, что в потомках будут чередоваться короткие строки особей-родителей.

4) Триадный кроссинговер.

Данная разновидность кроссинговера отличается от однородного тем, что после отбора пары родителей из остальных членов популяции случайным образом выбирается особь, которая в дальнейшем используется в качестве маски. Далее 10% генов маски мутируют. Затем гены первого родителя сравниваются с генами маски: если гены одинаковы, то они передаются первому потомку, в противном случае на соответствующие позиции хромосомы потомка переходят гены второго родителя. Генотип второго потомка отличается от генотипа первого тем, что на тех позициях, где у первого потомка стоят гены первого родителя, у второго потомка стоят гены второго родителя и наоборот.

5) Перетасовочный кроссинговер.

В данном алгоритме особи, отобранные для кроссинговера, случайным образом обмениваются генами. Затем выбирают точку для одноточечного кроссинговера и проводят обмен частями хромосом. После скрещивания созданные потомки вновь тасуются. Таким образом, при каждом кроссинговере создаются не только новые потомки, но и модифицируются родители (старые родители удаляются).

.

6) Кроссинговер с уменьшением замены или ограниченный кроссинговер .

Оператор уменьшения замены ограничивает кроссинговер, чтобы всегда, когда это возможно, создавать новые особи. Это осуществляется за счет ограничения на выбор точки разреза: точки разреза должны появляться только там, где гены различаются.

Как было показано выше, кроссинговер генерирует новое решение (в виде особи-потомка) на основе двух имеющихся, комбинируя их части. Поэтому число различных решений, которые могут быть получены кроссинговером при использовании одной и той же пары готовых решений, ограничено. Соответственно, пространство, которое ГА может покрыть, используя лишь кроссинговер, жестко зависит от генофонда популяции. Чем разнообразнее генотипы решений популяции, тем больше пространство покрытия.

При обнаружении локального оптимума соответствующий ему генотип будет стремиться занять все позиции в популяции, и алгоритм может сойтись к ложному оптимуму. Поэтому в генетическом алгоритме важную роль играют мутации. Существует несколько способов мутирования генов. Вопрос о выборе подходящего оператора мутации решается в рамках поставленной задачи.

3.3.2 Рекомбинация действительных значений.

При этом хромосома представляется действительным (вещественным) числом.

1) Дискретная рекомбинация

1-й родитель	12	25	5
2-й родитель	123	4	34
1-й образец (маска)	2	2	1
2-й образец (маска)	1	2	1
1-й потомок	123	4	5
2-й потомок	12	4	5

Рис.3.10

Этот вид скрещивания определен над векторами, компонентами которых являются вещественные числа. Рассмотрим на примере двух особей, соответствующие вектора которых имеют три целых значений. Для каждой переменной (позиции) значение переменной выбирается из первого или второго родителя с равной вероятностью. Пример выполнения этого оператора показан на рис.3.10.

Здесь образцы (маски) показывают принадлежность данной компоненты потомка определенному родителю. Отметим, что этот тип рекомбинации применим к значениям любого типа.

2) Промежуточная рекомбинация.

Этот метод применим только для особей, представленных только вещественными значениями. Здесь значения потомков строятся в окрестности или между значениями родителей. Потомок формируется следующим образом:

$$Of = P_1 + \alpha \cdot (P_2 - P_1)$$
$$\alpha \in [-d, d+1],$$

где P_1, P_2 – вещественные значения, представляющие первого и второго родителя;

Of – вещественное значение, представляющее потомка;

α - масштабирующий множитель, который выбирается случайно из отрезка [-d, 1+d].

Для обычной промежуточной рекомбинации $d = 0$ и $\alpha \in [0,1]$.

Для обобщенной промежуточной рекомбинации $d > 0$, обычно принимают $d = 0{,}25$.

1-й родитель	12	25	5
2-й родитель	123	4	34
Случайно выбраны следующие значения коэффициента α			
1-й образец	0,5	1,1	0,1
2-й образец	0,1	0,8	0,5
1-й потомок	67,5	1,9	2,1
2-й потомок	23,1	8,2	19,5

Рис.3.11

Значение каждой переменной, в том числе и для векторов формируется по приведенному выражению. Отметим, что здесь используется уже чисто арифметические операции (сложение и умножение). Этот оператор совершенно не похож на классический кроссинговер. Фактически, этот оператор заимствован из другого направления эволюционных вычислений – «эволюционные стратегии». На рис.3.11 показан пример выполнения этого оператора для тех же данных, которые использовались в предыдущем примере.

Для примера подробно рассмотрим выполнение оператора для 1-ой компоненты $OF = P_1 + \alpha \cdot (P_2 - P_1) = 12 + 0.5 \cdot (123 - 12) = 67.5$.

3)Линейная рекомбинация

Этот вид оператора аналогичен предыдущему, за исключением того, что значение масштабирующего множителя α одинаковы для всех переменных (компонент) векторов. Пример выполнения этого оператора представлен на рис.3.12.

1-й родитель	12	25	5
2-й родитель	123	4	34
Случайно отобраны следующие значения коэффициента α			
1-й образец		$\alpha = 0,5$	
2-й образец		$\alpha = 0,1$	
1-й потомок	67,5	14,5	19,5
2-й потомок	23,1	22,9	7,9

Рис.3.12.

3.4 Оператор мутации

После выполнения операторов рекомбинации (скрещивания) полученные потомки с вероятностью P_m подвергаются мутации, которая может быть выполнена различными способами.

3.4.1 Двоичная мутация

1) Классическая мутация

Этот вид оператора мы уже рассматривали в простом ГА. Здесь для каждой особи случайно выбирается позиция и с малой вероятностью от P_m=0,01 до P_m=0,001 выполняется инвертирование значения переменной в выбранной позиции. Пример выполнения этого оператора представлен на рис.3.13.

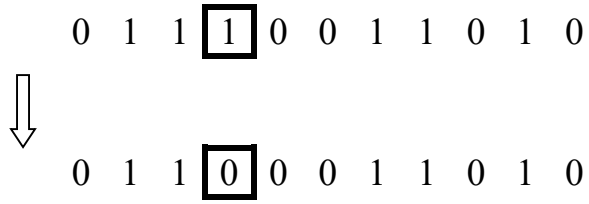

Рис.3.13.

2) Оператор инверсии

Иногда используется следующий оператор инверсии, фактически являющийся разновидностью мутации. При этом случайным образом выбираются 2 позиции в особи и далее производится обмен значениями генов между ними. Пример выполнения этого оператора представлен на рис.3.14.

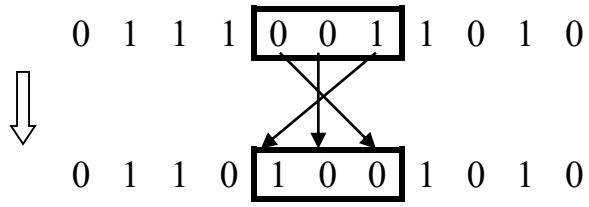

Рис.3.14.

3.4.2 Мутация над вещественными числами

Мутация над вещественными потомками выполняется путем сложения особи с небольшим случайным значением, которое называется шагом мутации. Выбор размера шага мутации зависит от рассматриваемой проблемы, и шаг в общем случае может изменяться в процессе решения задачи. Маленький шаг дает большую точность, но ведет к большим временным затратам. Мутация с постоянным шагом и постоянной вероятностью называется однородной.

1) Однородная мутация

Оператор мутации над вещественным числом выполняется следующим образом

$$V_m = v \pm r \cdot \Delta,$$

где V, V_m – значения вещественной переменной до и после мутации;

знаки + или − выбираются с равной вероятностью,

r = 0,5 ×<модуль изменения переменной V>

Однородная мутация над вещественными числами выполняется по формуле:

$$\Delta = \sum a(i) \cdot 2^{-i} \quad a(i) = \begin{cases} 1 & if \quad P_1 = \frac{1}{m} \\ 0 & if \quad P_0 = 1 - \frac{1}{m} \end{cases},$$

где m — параметр.

Новая особь, получившаяся при такой мутации, в большинстве случаев не намного отличается от старой. Это связано с тем, что вероятность маленького шага мутации выше, чем вероятность большого шага. При m = 20, данный алгоритм мутации пригоден для локализации оптимума с точностью $r \cdot 2^{-19}$.

Часто для повышения эффективности поиска вероятность мутации и шаг изменяются в процессе решения задачи. Мутация с постоянной вероятностью может привести как к увеличению, так и к уменьшению значения целевой функции. На этапе сходимости ГА к оптимуму целесообразно уменьшать вероятность случайной мутации. Обычно на начальном этапе $P_m = 0.05 \ldots 0.1$. А на конечном этапе вероятность мутации уменьшают. Для реализации этой процедуры иногда используют метод моделирования отжига (simulation annealing), который дает следующий закон изменения вероятности мутации:

$$P_m = P_m^0 \cdot e^{-\frac{1}{t}},$$

где t – номер поколения.

2) Неоднородная *мутация*

Здесь изменяется шаг мутации. В начале шаг мутации имеет достаточно большое значение, которое далее постепенно уменьшается. Рассмотрим этот тип оператора для векторного случая

$$S_v^t = \langle V_1, V_2, \ldots, V_k, \ldots V_m \rangle$$
$$V_k \in [l_k, U_k]$$

.

В результате выполнения мутации получаем следующий вектор (особь популяции)

$$S_v^{t+1} = \langle V_1, V_2, \ldots, V_k^{'}, \ldots V_m \rangle$$
$$k \in [1, \ldots m]$$

,

где компонента вектора вычисляется следующим образом

$$V'_k = \begin{cases} V_k + \Delta(t, U_k - V_k) & \textit{при} \quad \textit{случ.} \quad \textit{число} = 0 \\ V_k - \Delta(t, V_k - l_k) & \textit{при} \quad \textit{случ.} \quad \textit{число} = 1 \end{cases},$$

где $\Delta(t, y) \in [0, y]$ и $\Delta(t, y)$ определяет шаг мутации, который с увеличением номера поколения t уменьшается.

$$\lim_{t \to \infty} \Delta(t, y) \to 0$$

Один из вариантов реализации функции, определяющей шаг $\Delta(t, y)$ следующий

$$\Delta(t, y) = y \cdot \left(1 - r^{\left(1 - \frac{t}{T}\right) \cdot b} \right),$$

где $r \in \{0,1\}$ (r-случайное число);

T – максимальное число поколений;

$b = 2$ – параметр, определяющий степень неоднородности.

На рис.3.15 показан для наглядности график изменения шага мутации, который в пределе стремится к 0.

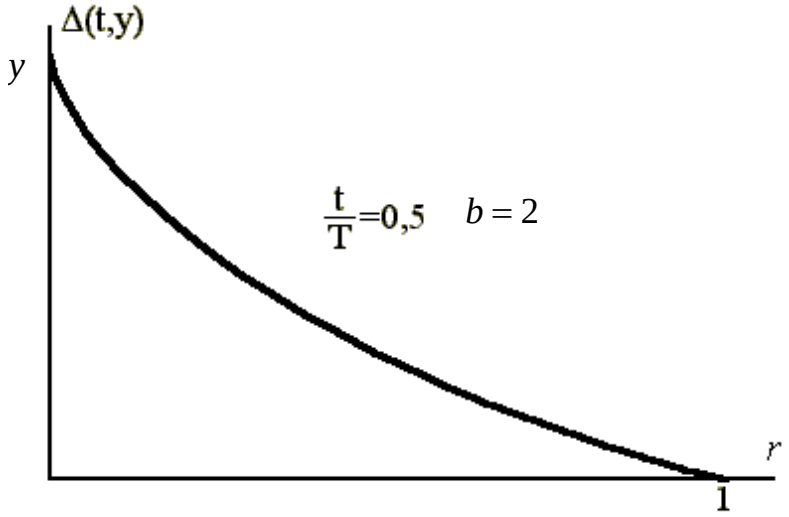

Рис.3.15 a)

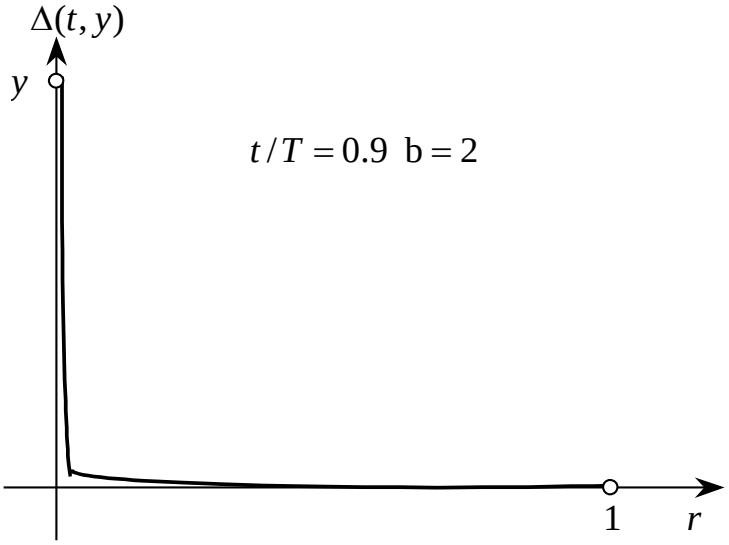

Рис.3.15 b)

3.4.2 Другие виды мутаций.

1. Присоединение случайного гена из совокупности всевозможных значений генов к концу последовательности хромосомы.

2. Вставка случайного гена из совокупности всевозможных значений генов в случайно выбранную позицию в последовательности хромосомы.

3. Удаление случайно выбранного гена из последовательности хромомсомы.

3.5 Сокращение промежуточной популяции

Необходимой компонентой является устранение плохих решений, полученных в процессе формирования новой популяции.

3.5.1 Глобальная редукция

Промежуточную популяцию (репродукционную группу) составляют все особи t-го поколения, особи полученные в результате скрещивания и мутации. Тогда численность этой популяции можно определить следующим образом

$$R^{t+1} = r^t + r_{cr}^t + r_m^t,$$

где: r^t - число особей предыдущей популяции;

r_{cr}^t – численность особей, полученных путем скрещивания;

r_m^t – число «мутантов».

Обычно в стационарных ГА мощность популяции поддерживается постоянной $N = |P(t)|$. Поэтому поскольку $R^{t+1} > N$, то необходимо устранить неудачные решения. Для этого существуют различные методы редукции.

1) Чистая замена

В простейшем случае с помощью скрещивания и мутации генерируются столько потомков, сколько было родителей. Далее родители устраняются, а потомки формируют следующее поколение $P(t+1)$.

При этом каждая особь живет лишь одно поколение. Такая схема часто используется в простом ГА. Однако при этом очевидно возможно, что некоторые очень хорошие решения могут быть заменены худшими, и лучшее решение будет потеряно.

2) Элитарная схема.

Создается промежуточная популяция, которая включает в себя как родителей, так и их потомков. Члены этой популяции оцениваются, а за тем из них выбираются N самых лучших (пригодных), которые и войдут в следующее поколение. Зачастую данный метод комбинируют с другими — выбирают в новую популяцию, например, 10 % «элитных» особей, а остальные 90 % — одним из традиционных методов селекции. Иногда эти 90 % особей создают случайно, как при создании начальной популяции перед запуском работы генетического алгоритма. Использование стратегии элитизма оказывается весьма полезным для эффективности ГА, так как не допускает потерю лучших

решений. К примеру, если популяция сошлась в локальном максимуме, а мутация вывела одну из строк в область глобального, то при предыдущей стратегии весьма вероятно, что эта особь в результате скрещивания будет потеряна, и решение задачи не будет получено. Если же используется элитизм, то полученное хорошее решение будет оставаться в популяции до тех пор, пока не будет найдено еще лучшее.

3) Отбор вытеснением.

В данном отборе выбор особи в новую популяцию зависит не только от величины ее пригодности, но и от того, есть ли уже в формируемой популяции особь с аналогичным хромосомным набором. Отбор проводится из числа родителей и их потомков. Из всех особей с одинаковой приспособленностью предпочтение сначала отдается особям с разными генотипами. Таким образом, достигаются две цели: во-первых, не теряются лучшие найденные решения, обладающие различными хромосомными наборами, во-вторых, в популяции постоянно поддерживается генетическое разнообразие. Вытеснение в данном случае формирует новую популяцию скорее из удаленных особей, вместо особей, группирующихся около текущего найденного решения.

4) Метод Больцмана, или метод отжига (Bolzman selection).

В данном методе вероятность отбора в новую популяцию зависит от управляющего параметра — температуры T. Обычно вероятность попадания в новую популяцию вычисляется по следующей формуле:

$$p = \frac{1}{1 + \exp \frac{f(i) - f(j)}{T}},$$

где f(i) и f(j) — значения целевой функции i и j особей, соответственно. Номера особей i и j выбираются случайно. Если значение p окажется больше случайного числа на интервале (0; 1), то в новую популяцию попадет особь f(i), иначе f(j).

В данном методе первым поколениям соответствуют высокие температуры, и вероятность отбора особей велика (поддерживается многообразие в новой популяции). Постепенно с ростом количества поколений

ГА температура снижается, вероятность отбора уменьшается и в новую популяцию попадают те особи, приспособленность которых минимальна.

5) Равномерная случайная замена.

При таком подходе потомков генерируется меньше, чем было родителей, далее случайно удаляется необходимое число родителей, которые заменяются новыми потомками.

В общем случае к репродукционной группе R^{t+1} может быть применен любой метод отбора родителей, которые были рассмотрены в разделе 3.2.

3.5.2 Локальная замена.

При локальном отборе особи выбираются из ограниченного окружения множества соседей. Замена особей потомками выполняется по такой же схеме. Таким образом сохраняется локальность информации. При этом используются те же структуры и отношения соседства, что и в разделе 2.4.

Родитель особи определяются первым родителем из локального окружения (множества соседей). Для выбора удаляемых родителей и отбора потомков для замены применяются следующие схемы:

- Ввести в новое поколение каждого потомка и заменить случайным образом особи из их окружения;
- Ввести каждого потомка и заменить худшей особью соответственного окружения;
- Ввести потомка лучше, чем худшая особь в окружении и заменить худшие особи.

4. Задачи комбинаторной оптимизации

До сих пор мы рассматривали применение ГА, в основном, при решении задачи численной оптимизации типа поиска экстремумов функции. Но в настоящее время ГА используются больше для решения задач комбинаторной оптимизации, и в этой области публикуется подавляющая часть научных работ.

Базовой задачей комбинаторной оптимизации является задача коммивояжера. Однако сначала мы рассмотрим задачи об укладке рюкзака и о покрытии, которые допускают использование простого (классического ГА).

4.1 Задача об укладке рюкзака

Эта задача имеет следующую неформальную простую постановку [4,5]. Имеется рюкзак объемом C и n различных предметов. Каждый предмет i имеет известный объем Wi и стоимость Pi ($i = 1,...,n$). В рюкзак можно положить целое число различных предметов. Нужно упаковать рюкзак так, чтобы полная стоимость уложенных предметов была максимальной, а их общий объем не превышал заданный объем C. Форма предметов здесь не учиты.

Формальная постановка задачи: для данного множества весов Wi , стоимостей Pi и объема C надо найти двоичный вектор

$X = (x_1, \ldots, x_n)$, где $x_i = \begin{cases} 1, & \text{если предмет помещается в рюкзак,} \\ 0 & \text{в противном случае,} \end{cases}$

и при этом должно выполняться условие:

$$V = \sum_{i=1}^{n} W_i \le C \text{ и } \sum P_i = max.$$

Так как решение задачи можно представить двоичным вектором X= (x_1,…, x_n), то очевидно при его поиске можно применить простой ГА со стандартными операторами скрещивания и мутации. Но при этом на каждом шаге итерации надо следить за тем, чтобы новые решения, полученные в результате скрещивания или мутации, удовлетворяли требуемому ограничению

$V \le C$. В случае невыполнения ограничения «неправильное» потенциальное решение должно быть уничтожено, что ведет к сокращению популяции.

В качестве фитнесс-функции в простейшем случае можно взять $P(X) = \sum_{i=1}^{n} x_i \cdot P_i$, но в этом случае, как указано выше, есть проблемы с неправильными решениями.

Данная задача относится к классу задач с ограничениями, при решении которых применяются следующие подходы [1,5,7]: 1) введение в фитнесс-функцию дополнительного штрафа; 2) использование алгоритмов “восстановления” некорректных решений.

1) В первом случае в фитнесс-функцию вводится дополнительная штрафная функция, которая для неправильных решений дает большие отрицательные значения ЦФ. При этом задача с ограничениями трансформируется в задачу без ограничений путем назначения штрафа для некорректных решений. Фитнесс-функция для каждой особи может быть определена следующим образом

$$f(X) = \sum_{i=1}^{n} x_i \cdot P_i - Pen(X) .$$

Разработано множество методов назначения штрафных значений, из которых ниже рассмотрено только три вида, когда рост значения штрафной функции относительно степени нарушения ограничения имеет логарифмический, линейный и квадратичный характер:

а) $Pen(X) = log_2(1 + \rho \cdot (\sum_{i=1}^{n} x_i \cdot W_i - C))$,

б) $Pen(X) = \rho \cdot (\sum_{i=1}^{n} x_i \cdot W_i - C)$,

в) $Pen(X) = (\rho \cdot (\sum_{i=1}^{n} x_i \cdot W_i - C))^2$,

Здесь для всех трех случаев $\rho = max_{i=1...n}\{ P_i / w_i \}$.

2) Второй подход к решению задач с ограничениями основан на специальных алгоритмах “восстановления” некорректных решений. Следует отметить, что многие алгоритмы восстановления требуют значительных вычислительных ресурсов, и полученные решения иногда требуют адаптации к конкретным практическим приложениям.

В этом случае в качестве фитнесс-функции используется $f(X') = \sum_{i=1}^{n} x_i' P_i$, где вектор X' – восстановленная версия исходного вектора X. Здесь следует отметить, по крайней мере, два аспекта. Во-первых, можно использовать различные алгоритмы восстановления. Во-вторых, восстановленные особи могут замещать только некоторую часть исходных особей в популяции. Процент замещаемых особей может варьироваться от 0% до 100% и его значение является важнейшим параметром метода восстановления. В некоторых работах отмечается, что наилучшие результаты получаются при 5%, во всяком случае, лучше, чем в двух крайних случаях – 0% (без замещения) и 100% (любая восстановленная особь заменяет исходную). Ниже приведен простой алгоритм восстановления.

```
Восстановление(X)
 {
  переполнение_рюкзака=false;
  X’=X;
  if ( f(X') = Σ_{i=1}^{n} x_i'·W_i > C )
     then переполнение_рюкзака=true;
 while(переполнение_рюкзака)
       {
         i=выбор предмета из рюкзака;
         // удаление выбранного предмета из рюкзака;
           x'_i = 0;
   if ( f(X') = Σ_{i=1}^{n} x'_i·W_i ≤ C )
          then переполнение_рюкзака=false;
        }
  }
```

При этом используются два основных способа выбора объекта:

а) случайный выбор объекта из рюкзака;

б) “жадное восстановление”, при котором вначале все предметы сортируются в порядке убывания их стоимости P_i и на каждом шаге для удаления выбирается предмет минимальной стоимости (из имеющихся в рюкзаке).

3) Третий подход к решению задач с ограничениями использует специальное отображение (декодирование) особей, которое гарантирует генерацию допустимого решения (с учетом ограничений), или используют проблемно-ориентированные генетические операторы, сохраняющие корректность решения.

Рассмотрим один из возможных вариантов алгоритма декодирования, который основан на кодировании решения вектором целых чисел, так называемом «упорядоченном представлении» (ordinal representation), подробно описанном в разделе 4.3.2. Здесь каждая хромосома кодируется вектором *e* целых чисел, где i-я компонента вектора – есть целое число в диапазоне от 1 до n-i+1. “Упорядоченное представление” использует для ссылок (базовый) список предметов L. Вектор *e* фактически содержит указатели на базовый список L. Декодирование вектора осуществляется путем выбора соответствующего предмета из текущего списка и удаления его из базового списка. Например, при базовом списке предметов L=(1,2,3,4,5,6) текущий вектор e=(4,3,4,1,1,1) декодируется в следующую последовательность предметов: 4, 3, 6, 1,2, 5. Первым в рюкзак включается четвертый элемент списка L - 4, который устраняется из L. Далее в рюкзак включается третий элемент из текущего списка L. Затем включается 6 - четвертый элемент текущего L и т.д. Подробнее описание этого представления и выполнение кроссинговера на нем описано в разделе 4.3.2. В данном методе хромосома может интерпретироваться как стратегия (порядок) включения предметов в решение. Отметим, что основным достоинством данного кодирования хромосомы является то, что на нем работает одноточечный кроссинговер. То есть для двух допустимых решений –

родителей кроссинговер порождает также допустимое решение–потомок. Оператор мутации при данном представлении выполняется путем замены i-го гена (целочисленной компоненты вектора) на случайное целое число из диапазона [1,…,n-i+1].

Алгоритм декодирования представлен ниже.

```
Декодирование(X)
 {
  генерация списка предметов L;
  i=1;
 Сумма_весов=0;
 Суммарная_стоимость=0;
  While(i ≤ n)
    {
      j= x_i;
      удаление j-го предмета из списка L;
      if(сумма _ весов + W_j ≤ C) then
          {
           Сумма_весов= Сумма_весов + W_j;
            Суммарная_стоимость= Суммарная_стоимость + P_j;
          }
       i=i+1;
     }
 }
```

Очевидно, что представленный алгоритм зависит от способа генерации списка предметов L. Обычно используются два метода генерации этого списка:

а) список предметов L генерируется в том порядке, в котором предметы расположены во входном файле (как правило, случайно);

б) список предметов L генерируется в порядке убывания их стоимостей (жадный алгоритм). Декодирование вектора X выполняется на основе отсортированного вектора. Например, x_{25} интерпретируется как 25-й предмет текущего списка L.

Задача об укладке рюкзака в различных вариантах имеет многочисленные практические приложения. Например, к ней можно свести оптимизацию загрузки транспорта и т.п.

4.2 Задача о покрытии

Есть множество элементов S и множество подмножеств $F(i)$, состоящих из элементов S. Необходимо найти минимальное число подмножеств из F таких, что объединение этих подмножеств содержит все элементы множества S.

Очевидно, здесь решение можно представить двоичным вектором

$$\overline{X} = (X(1)\ldots X(n)),$$

где $X(i) = 1$ если $F(i)$входит в покрытие;

$X(i) = 0$ если $F(i)$входит в покрытие;

При этом:

$$\cup X(i)F(i) = S$$
$$\sum X(i) = \min$$

Очевидно, что для решения этой задачи можно также использовать простой ГА со стандартными операторами кроссинговера и мутации.

4.3 Задача коммивояжера

Напомним постановку этой классической комбинаторной оптимизационной задачи. Коммивояжер должен посетить каждый город на своём участке один раз и вернуться в исходную точку, как это показано, например, на рис.4.1.

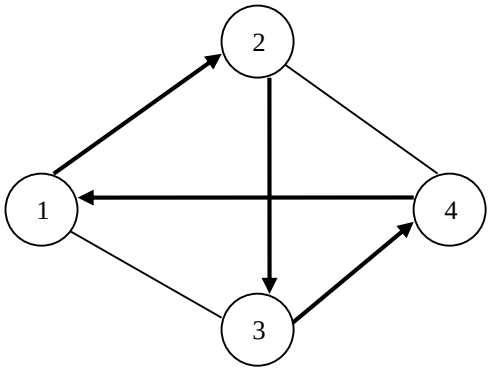

Рис.4.1.

Дана стоимость проезда между каждой парой городов и необходимо выбрать оптимальный маршрут с min стоимостью тура (обхода всех городов). Пространством поиска решений этой задачи является множество перестановок городов n!. Любая простая (одиночная) перестановка n городов даёт решение, являющееся полным туром из n городов. Оптимальным решением является перестановка, которая даёт min стоимостью тура. Известно, что эта задача является NP – полной, т.е. переборного типа. Она имеет многочисленные практические приложения, в которых число “городов” может быть достаточно большим. Например, при производстве плат проблема сверления переходных отверстий имеет ~ 1700 “городов”, при производстве СБИС возникают задачи с числом “городов” ~ 1,0 млн. и т.п. За последние десятилетия разработано достаточно много алгоритмов решения этой задачи, дающих субоптимальное решение. В последнее десятилетие эта задача является базовой для исследования ГА в области комбинаторной оптимизации.

Очевидно, что двоичное представление тура при решении ЗК не целесообразно. Действительно если мы интересуемся оптимальной перестановкой городов, т.е. (i_1,i_2, … , i_n) и используем двоичное представление в виде одного бинарного вектора, то изменение даже в одном бите может дать двоичный вектор, не принадлежащий к области решения, т.е. не являющейся перестановкой n городов.

В последнее время, в основном, используется три способа представления тура при решении ЗК с использованием ГА: 1) представление соседства; 2) представление порядка; 3) представление путей. Для каждого из этих представлений разработаны свои “генетические” операторы. Операторы мутации относительно легко определить на этих представлениях с помощью одиночной перестановки пары городов. Поэтому в дальнейшем мы, в основном, рассмотрим операторы кроссинговера.

4.3.1. ***Представление соседства.***

При этом тур представляется списком соседних городов в виде вектора целых чисел. Город j находиться в позиции i если и только если в туре после города i посещается город j, как это показано на рис.4.2.

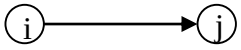

Рис.4.2

Например, вектор – список соседства (2 4 8 3 9 7 1 5 6) представляет следующий тур 1-2-4-3-8-5-9-6-7.

Любой тур имеет единственное представление списком соседства. Однако некоторые "списки соседства" могут представлять “неправильные” туры. Например, список соседства (2 4 8 1 9 3 5 7 6) представляет “частичный” тур 1-2-4-1, который охватывает только три города. Следовательно представление списком соседей не поддерживает классический оператор кроссинговера, так как при перестановке частей списков могут получиться неправильные, “частичные”, туры. Поэтому необходим некоторый алгоритм восстановления полного тура.

Для данного представления были предложены и исследованы 3 основных оператора кроссинговера: а) обмен ребер (дуг) графа; б) обмен подтуров; в) эвристический кроссинговер. Далее рассмотрим их более детально.

1) Обмен ребер

Этот тип кроссинговера строит потомок случайным выбором ребра (пары городов i - j) из первого родителя, затем выбирает соответствующее ребро из второго родителя и т.д. Оператор наращивает тур выбором ребер из различных родителей. Если новое ребро (из одного из родителей) образует преждевременный (частичный) цикл в текущий (ещё не полный) тур, то оператор выбирает (случайно) вместо этого ребра одно из оставшихся ребер, которое не дает преждевременного цикла.

Например, для 2-х родителей

P_1 = (2 3 8 7 9 4 1 5 6) 1-2-3-8-5-9-6-4-7-1

1-й обмен ↕ ↕ ← 2-й обмен (неудачный)

P_2 = (7 5 1 6 9 2 8 4 3) 1-7-8-4-6-2-5-9-3-1

Получаем поэтому (случайно, выбрана 6 из оставшихся (6, 4, 3))

σ_1 = (2 5 8 7 9 1 6 4 3) 1-2-5-9-3-8-4-7-6-1

Здесь выполнен (случайный) обмен 3 $\rightarrow$ 5 во второй позиции. Затем при попытки (случайного) обмена в 7-й позиции 8 $\rightarrow$ 1 получается преждевременный цикл и случайно выбирается 6 (одна из оставшихся 6, 4, 3) и т.о.

2*) Обмен подтуров*

Этот оператор строит потомок путём выбора (случайной длины) подтура из I-го родителя, затем выбора подтура из второго родителя и их обмена. Как и ранее в случае преждевременного цикла оператор случайно выбирает другое ребро (город /из оставшихся/ не вошедших в построенный тур).

3) Эвристическое скрещивание

Этот оператор строит потомка случайным выбором города в качестве начальной точки для формирования тура. Затем сравниваются два ребра, исходящие из этого города, имеющихся в двух родителях, и из них выбирается лучшее с меньшей стоимостью. Полученный город (в который входит выбранное ребро на предыдущем этапе), используется в качестве начальной точки при выборе следующего ребра и т.д. Как и ранее в случае возникновения преждевременного цикла, следующий город выбирается случайно из городов, не вошедших в текущий тур.

Известна следующая модификация этого оператора.

1) Если оптимальное (с меньшей стоимостью) ребро дает цикл в потомках, то проверяется альтернативное ребро (с большей стоимостью) на генерацию преждевременного цикла. Если это ребро не генерирует цикл, то оно включается в тур, иначе выбирается min из q (параметр) случайно выбранных из оставшихся городов.

Преимущество этого представления в том, что анализ схем (шаблонов) можно проводить по аналогии с двоичным случаем. Например, схема (* * * 3 * 7 * * *) представляет множество всех туров с ребрами (4, 3) и (6, 7). Однако основной недостаток этого представления в весьма посредственных результатах тестовых задач для всех 3-х рассмотренных операторов. Кроссинговер обмена ребер часто разрывает хорошие туры при обмене ребер родителей. Кроссинговер обмена подтуров даёт лучшие результаты, чем предыдущий т.к. “отношение разрыва” здесь меньше. Эвристический оператор даёт лучшие результаты, т.к. первые 2 оператора не учитывают стоимости ребер. С другой стороны, эвристический оператор выбирает лучшее ребро из 2-х возможных и поэтому даёт лучшие результаты. Однако экспериментальные данные показывают тоже “не выдающиеся” результаты.

4.3.2 *Упорядоченное представление*

Представляет тур как список из n – городов i-й элемент списка имеет номер от 1 до n-i+1. При этом существует упорядоченный список городов c, который служит для ссылок из упорядоченного представления.

Рассмотрим на конкретном примере тура 1-2-4-3-8-5-9-6-7, для которого упорядоченный список просто c = (1 2 3 4 5 6 7 8 9). Тогда данный тур при этом упорядоченном списке представляется следующим списком *e* ссылок. *e* = (1 1 2 1 4 1 3 1 1) список *e* интерпретируется следующим образом.

Первый номер из списка *e* равен 1, поэтому берём в тур 1-й город из списка *c*, и удаляем его из c. Тогда:

Тур 1 и c = (2 3 4 5 6 7 8 9) e = (1 1 2 1 4 1 3 1 1)
↓

Следующий номер списка e также равен 1, поэтому мы берем и удаляем 1-й город из текущего списка c и добавляем его в тур и передвигаем ↓ по e. Имеем

Тур 1-2 и c = (3 4 5 6 7 8 9) e = (1 1 2 1 4 1 3 1 1)
↓

Следующий номер списка e равен 2, поэтому мы берем и удаляем 2-й город из текущего списка c и добавляем его в тур и передвигаем ↓ по e. Имеем

Тур 1-2-4, c = (3 5 6 7 8 9) e = (1 1 2 1 4 1 3 1 1)
↓

Следующий номер в e равен 1, поэтому мы берем в тур и удаляем 1-й город из c текущего c

Тур 1-2-4-3 c = (5 6 7 8 9) e = (1 1 2 1 4 1 3 1 1)
↓

Следующий номер в e равен 4, поэтому берем 4-й номер из c.

Тур 1-2-4-3-8 c = (5 6 7 9) e = (1 1 2 1 4 1 3 1 1)
↓

Далее берём 1-й номер из c

Тур 1-2-4-3-8-5 c = (6 7 9) e = (1 1 2 1 4 1 3 1 1)
↓

Далее 3-й номер из c

Тур 1-2-4-3-8-5-9 c = (6 7) e = (1 1 2 1 4 1 3 1 1)
↓

Аналогично берём 1-й номер из c и имеем

Тур 1-2-4-3-8-5-9-6 c = (7) e = (1 1 2 1 4 1 3 1 1)
↓

Последний номер равен 1; получаем в итоге тур 1-2-4-3-8-5-9-6-7.

Основное преимущество упорядоченного представления в том, что классический кроссинговер работает!

Например, для родителей

P_1 = (1 1 2 1 | 4 1 3 1 1) и

P_2 = (5 1 5 5 | 5 3 3 2 1), которые соответствуют турам

1-2-4-3-8-5-9-6-7 и

5-1-7-8-9-4-6-3-2

имеем следующих потомков при скрещивании

O_1 = (1 1 2 1 5 3 3 2 1)

O_2 = (5 1 5 5 4 1 3 1 1), которые представляют туры

1-2-4-3-9-7-8-6-5 и

5-1-7-8-6-2-9-3-4

Очевидно, что частичные туры слева от точки кроссинговера не изменяются, в тоже время частичные туры справа от точки скрещивания рвутся случайным образом. Машинные эксперименты этого представления с использованием классического кроссинговера показывают удовлетворительные результаты.

4.3.3 *Представление путей*

Это представление возможно самое естественное для тура. Например, тур 5-1-7-8-9-4-6-2-3 предоставляется просто упорядоченным списком (5 1 7 8 9 4 6 2 3) городов, входящих в тур.

Совсем недавно для этого предоставления были определены и исследованы 3 типа кроссинговера:

1) частично соответствующей ОК (partially-mapped PMX);

2) упорядоченный ОК (order OX)

3) циклический ОК (cycle CX)

1) частично соответствующий ОК (PMX)

Строит потомок путем выбора подпоследовательности тура из одного родителя и сохранения порядка и позиции городов из другого родителя насколько это возможно. Подпоследовательность из тура выбирается случайно с помощью двух “секущих” точек, которые служат границами для операции обмена. Например, для родителей

P1=(1 2 3 | 4 5 6 7| 8 9) и

P2 = (4 5 2 | 1 8 7 6 | 9 3)

Потомок строится так.

Производим обмен выделенными подтурами и получаем:

T1 = (X X X | 1 8 7 6 | X X)

T2 = (X X X | 4 5 6 7 | X X)

Этот обмен определяет также отображение 1 ⟷ 4 , 8 ⟷ 5,

7 ⟷ 6, 6 ⟷ 7.

Затем заполняем (вместо ‘X’) города из исходных родителей, для которых нет конфликтов (не образуются преждевременный цикл). T1 = (X 2 3 | 1 8 7 6 | X 9)

T2 = (X X 2 | 4 5 6 7 | 9 3)

Далее первый ‘X’ в O_1 – 1 заменяем на 4, так как имеется отображение 1 ⟷ 4. Аналогично второй ‘X’ в потомке O_1 заменяется на 5, и в O_2 ‘X’ и ‘X’ заменяются на 1 и 8. В результате получаем

O_1 = (4 2 3 | 1 8 7 6 | 5 9)

O_2 = (1 8 2 | 4 5 6 7 | 9 3)

Кроссинговер PMX использует подобие (схожесть) по значениям и порядку одновременно

2) упорядоченный ОК (OX) строит потомок выбором подтура из одного родителя с сохранением относительно порядка городов из другого родителя. Например, для родителей

P1 = (1 2 3 | 4 5 6 7 | 8 9)

P2 =(4 5 2 | 1 8 7 6 | 9 3)

Потомок строится так.

Сначала сегменты между двумя секущими точками копируются в потомки:

O_1 = (X X X | 4 5 6 7 | X X)

O_2 = (X X X | 1 8 7 6 | X X)

Далее, начиная со второй секущей точки одного родителя города из другого родителя копируется в том же порядке, пропуская уже присутствующие в

построенном подтуре. По достижению конца стринга мы продолжаем этот процесс с первой позиции и до первой точки сечения (по кольцу).

После второй точки сечения во втором родителе мы имеем следующую последовательность:

9 – 3 – 4 – 5 – 2 – 1 – 8 – 7 – 6, удалив из нее 4, 5, 6, 7 поскольку они уже есть в первом родителе, получаем последовательность 9 – 3 – 2 – 1 – 8. Ее мы помещаем в первый потомок, начиная со второй точки сечения (по кольцу) и получаем

T1 = (2 1 8 | 4 5 6 7 | 9 3).

Аналогично получаем

T2 = (3 4 5 | 1 8 7 6 | 9 2).

Оператор кроссинговера OX использует то, что при представлении тура путем прежде всего важен порядок, т. е. например два тура 9 – 3 – 4 – 5 – 2 – 1 – 8 – 7 – 6 и 4 – 5 – 2 – 1 – 8 – 7 – 6 – 9 – 3 идентичны.

3) Циклический ОК (CX) строит потомки таким образом, что каждый город вместе со своей позицией идет от одного из родителей. Например, для

P_1 = (1 2 3 4 5 6 7 8 9) и
P_2 = (4 1 2 8 7 6 9 3 5)

сначала получаем путем выбора первого города из родителя O1= (1X X X X X X X X). Выбор следующего города определяет текущая позиция второго родителя. В нашем примере это город 4, что дает O1= (1 X X 4 X X X X X). Город 4 в свою очередь имплицирует город 8 (из второго родителя), что дает O1= (1 X X 4 X X X 8 X).

Аналогично получаем города 3, 2 в O1= (1 2 3 4 X X X 8 X). Здесь мы вынуждены прервать этот процесс, так как выбор $2 \to 1$ ведет к преждевременному циклу. Поэтому оставшиеся города берутся из другого родителя P2 (с сохранением порядка) и в результате получаем потомковO1 = (1 2 3 4 7 6 9 8 5) и O2 = (4 1 2 8 5 6 7 3 9). Таким образом, оператор ОК CX сохраняет абсолютные позиции потомков и родителей.

Оператор ОК CX сохраняет абсолютные позиции потомков и родителей.

Список литературы

1. Z.Michalevich Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Third, revised and extended edition, 1999. – 387 p.
2. D.E.Goldberg Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989. – 215 p.
3. J.H.Holland Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. University of Michigan, 1975. – 210 p.
4. В.М.Курейчик Генетические алгоритмы и их применение. Таганрог: Изд-во ТРТУ, издание второе, дополненное, 2002. – 242 с.
5. Скобцов Ю.А. Основы эволюционных вычислений.- Донецк: ДонНТУ, - 2008.- 326 с.
6. Скобцов Ю.А., Сперанский Д.В. Эволюционные вычисления: учебное пособие.-М.:Национальный Открытый Университет «ИНТУИТ» 2015.- 331с.