1.2. Генетические алгоритмы

Как отмечалось в предисловии, оптимизация является важнейшим этапом решения задач идентификации. Основные трудности применения классических методов оптимизации нелинейных функций [10,58] связаны с проблемами локального экстремума (рис. 1.3) и <проклятия размерности> (рис. 1.4).

Рис. 1.3. Проблема локального экстремума

Рис. 1.4. Проблема «проклятия размерности»

Попытки преодоления указанных проблем привели к созданию теории генетических алгоритмов, которые выращивают оптимальное решение путем скрещивания исходных вариантов с последующей селекцией по некоторому критерию (рис. 1.5). Излагаемые в этом разделе общие сведения о генетических алгоритмах, базируются на работах [8,58,59,81]. Примеры, иллюстрирующие генетические алгоритмы, являются оригинальными.

Рис. 1.5. Идея генетического алгоритма (Goldberg D. Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989)

1.2.1. Принципы построения генетических алгоритмов

Генетический алгоритм (ГА) можно рассматривать как одну из разновидностей случайного поиска [30], которая основана на механизмах, напоминающих естественный отбор и размножение.

В отличие от существующих методик, ГА начинает работу с некоторого случайного набора исходных решений, который называется *популяцией*. Каждый элемент из популяции называется *хромосомой* и представляет некоторое решение проблемы в первом приближении. Хромосома представляет собой строку символов некоторой природы, не обязательно бинарных. Хромосомы эволюционируют на протяжении множества итераций, носящих название *поколений* (или генераций). В ходе каждой итерации хромосома оценивается с использованием некоторой меры соответствия (англ. fitness function), которую мы будем называть *функцией соответствия*. Для создания следующего поколения новые хромосомы, называемые *отпрысками*, формируются либо путем скрещивания (англ. crossover) двух хромосом - родителей из текущей популяции, либо путем случайного изменения (*мутации*) одной хромосомы. Новая популяция формируется путем (а) выбора согласно функции соответствия некоторых родителей и отпрысков и (б) удаления оставшихся для того, чтобы сохранять постоянным размер популяции.

Хромосомы с большей функцией соответствия имеют больше шансов быть выбранными (выжить). После нескольких итераций алгоритм сходится к лучшей хромосоме, которая является либо оптимальным, либо близким к оптимальному решением. Пусть $P(t)$ и $C(t)$ являются родителями и отпрысками из текущей генерации t . Общая структура генетического алгоритма (рис. 1.6) имеет вид:

Процедура: Генетический алгоритм

begin

t :=0 ;

Задать_начальное_значение $P(t)$;

Оценить $P(t)$ с помощью функции соответствия;

while (нет условия_завершения) **do**

Скрещивать $P(t)$ чтобы получить $C(t)$;

Оценить $C(t)$ с помощью функции соответствия;

Выбрать $P(t+1)$ из $P(t)$ и $C(t)$;

$t := t+1$;

end

end.

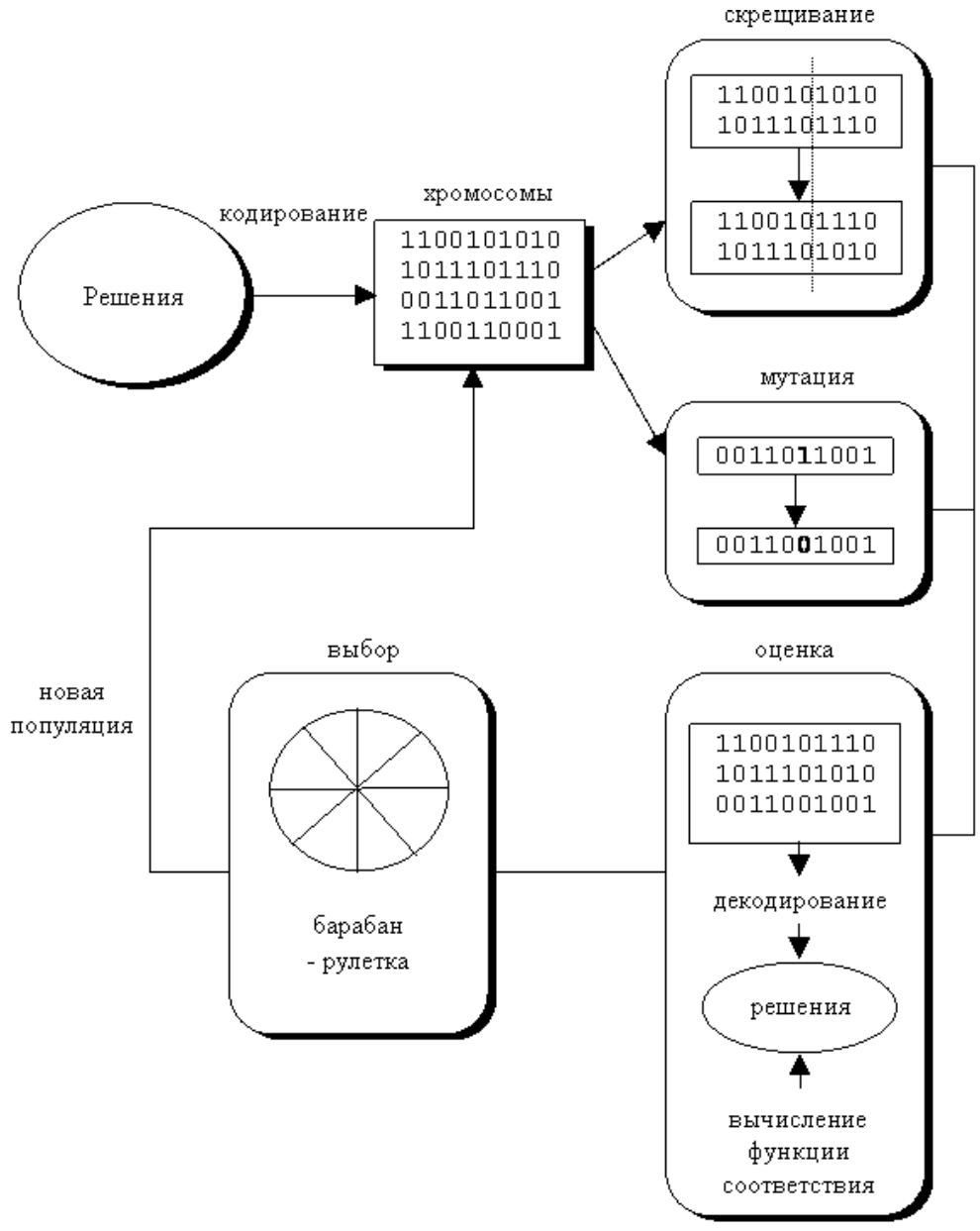


Рис. 1.6. Обобщенная структура генетического алгоритма

Таким образом, используются два вида операций:

1. Генетические операции: скрещивание и мутация;

2. Эволюционная операция: выбор.

Генетические операции напоминают процесс наследования генов при создании нового отпрыска в каждой генерации. Эволюционная операция, осуществляющая переход от одной популяции к следующей, напоминает процесс Дарвиновской эволюции.

1.2.2. Основные операции генетических алгоритмов

Операция скрещивания. *Скрещивание* является главной генетической операцией. Эта операция выполняется над двумя хромосомами- родителями и создает отпрыск путем комбинирования особенностей обоих родителей. Приведем простейший пример скрещивания. В начале выберем некоторую

случайную точку (точка скрещивания - англ. cut-point), после этого создадим хромосому-отпрыск путем комбинирования сегмента первого родителя, стоящего слева от выбранной точки скрещивания, с сегментом второго родителя, стоящего по правую сторону от точки скрещивания, как это показано на рис. 1.7.

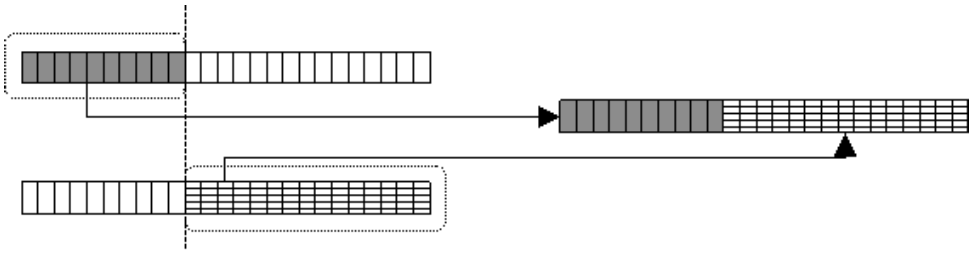


Рис. 1.7. Операция скрещивания

Этот метод работает очень хорошо, если хромосомы представляют собой битовые строчки. Кроме того производительность всего генетического алгоритма в первую очередь зависит от производительности используемой
операции скрещивания.

Доля производимых на каждой итерации отпрысков называется *коэффициентом скрещивания* (p_c). Произведение p_c **x** *размер_популяции* показывает количество отпрысков. Большое значение этого коэффициента позволяет исследовать больше областей пространства поиска (или пространства решений) и уменьшает шанс попадания в локальный минимум. Но если значение p_c слишком велико, то это приведет к большим затратам времени вычислений на исследование бесперспективных областей.

Операция мутации. *Мутация* - это фоновая операция, производящая случайное изменение в различных хромосомах. Наипростейший вариант мутации состоит в случайном изменении одного или более генов. В ГА мутация играет важную роль для (а) восстановления генов, выпавших из популяции в ходе операции выбора, так что они могут быть опробованы в новых комбинациях, (б) формирования генов, которые не были представлены в исходной популяции. Интенсивность мутаций определяется *коэффициентом мутаций* (p_m). Он представляет собой долю генов, подвергающихся мутации на данной итерации, в расчете на их общее число. Слишком малое значение этого коэффициента приводит к тому, что многие гены, которые могли бы быть полезными, никогда не будут рассмотрены. В то же время слишком большое значение коэффициента p_m приведет к большим случайным возмущениям. Отпрыски перестанут быть похожими на родителей и алгоритм потеряет возможность обучаться, сохраняя наследственные признаки .

1.2.3. Стратегии поиска

Поиск является одним из наиболее универсальных методов нахождения решения для случаев, когда априори не известна последовательность шагов, ведущая к оптимуму.

Существуют две поисковые стратегии: эксплуатация наилучшего решения и исследование пространства решений. Градиентный метод является примером стратегии, которая выбирает наилучшее решение для возможного улучшения, игнорируя в то же время исследование всего пространства поиска. Случайный поиск является примером стратегии, которая, наоборот, исследует пространство решений, игнорируя исследование перспективных областей поискового пространства. Генетический алгоритм представляет собой класс поисковых методов общего назначения, которые комбинируют элементы обоих стратегий. Использование этих методов позволяет удерживать приемлемый баланс между исследованием и эксплуатацией наилучшего решения. В начале работы генетического алгоритма популяция случайна и имеет разнообразные элементы. Поэтому оператор скрещивания осуществляет обширное исследование пространства решений. С ростом значения функции соответствия получаемых решений оператор скрещивания обеспечивает исследование окрестностей каждого из них. Другими словами, тип поисковой стратегии (эксплуатация наилучшего решения или исследование области решений) для оператора скрещивания определяется разнообразием популяции, а не самим этим оператором.

1.2.4. Отличие от классического поиска

В общем, алгоритм решения оптимизационных проблем представляет собой последовательность вычислительных шагов, которые асимптотически сходятся к оптимальному решению. Большинство классических методов оптимизации генерируют детерминированную последовательность вычислений, основанную на градиенте или производной целевой функции более высокого порядка. Эти методы применяются к одной исходной точке поискового пространства. Затем решение постепенно улучшается в направлении наискорейшего роста или убывания целевой функции. При таком поточечном подходе существует опасность попадания в локальный оптимум.

Генетический алгоритм осуществляет одновременный поиск по многим направлениям путем использования популяции возможных решений. Переход от одной популяции к другой позволяет избежать попадания в локальный оптимум. Популяция претерпевает нечто наподобие эволюции: в каждом поколении относительно хорошие решения репродуцируются, в то время как относительно плохие отмирают. ГА используют вероятностные правила для определения репродуцируемой или уничтожаемой хромосомы, чтобы направить поиск к областям вероятного улучшения целевой функции.

1.2.5. Преимущества генетических алгоритмов

Существуют два главных преимущества генетических алгоритмов перед классическими оптимизационными методиками:

1. ГА не имеет значительных математических требований к видам целевых функций и ограничений. Исследователь не должен упрощать модель объекта, теряя ее адекватность, и искусственно добиваясь возможности применения доступных математических методов. При этом могут использоваться самые разнообразные целевые функции и виды ограничений (линейные и нелинейные), определенные на дискретных, непрерывных и смешанных универсальных множествах.

2. При использовании классических пошаговых методик глобальный оптимум может быть найден только в том случае когда проблема обладает свойством выпуклости. В тоже время эволюционные операции генетических алгоритмов позволяют эффективно отыскивать глобальный оптимум.

1.2.6. Терминология

Поскольку ГА происходят как из естественных наук (генетика), так и из компьютерных наук, то используемая терминология представляет собой сплав естественного и искусственного. Соответствие терминов, относящихся к ГА и тех, которые относятся к решению оптимизационных проблем, приведено в табл. 1.1.

Таблица 1.1

Основные термины теории ГА

ГА	Объяснение

1. Хромосомы	Решение (код)
2. Ген (несколько бит)	Часть решения
3. Локус (местоположение)	Позиция гена в хромосоме
4. Аллель	Значение гена
5. Фенотип	Раскодированное решение
6. Генотип	Закодированное решение

1.2.7. Примеры генетической оптимизации

В этом разделе мы подробно объясним как в действительности работает генетический алгоритм на двух простых примерах.

Пример 1.9. Задача оптимизации. Рассмотрим нелинейную целевую функцию с ограничениями:

$$f(x_1, x_2) = \left(-2x_2^3 + 6x_2^2 + 6x_2 + 10\right) \cdot \sin\left(\ln(x_1) \cdot e^{x_2}\right)$$

$0.5 \le x_1 \le 1.1$, $1.0 \le x_2 \le 4.6$

Требуется найти: $\max_{x_1, x_2} f(x_1, x_2)$

Поверхность целевой функции показана на рис. 1.8.

Кодирование. Для реализации генетического алгоритма необходимо закодировать оптимизируемые параметры в двоичные строчки. Длина строчки зависит от требуемой точности. Например, пусть переменная x_j имеет интервал изменения $[a_j, b_j]$, и требуемая точность - пять знаков после запятой. В этом случае интервал изменения переменной x_j должен быть разбит как минимум на $(b_j - a_j) \times 10^5$ квантов.

Требуемое число битов m_j находится по формуле:

$$2^{m_j - 1} < (b_j - a_j) \times 10^5 \le 2^{m_j} - 1$$

Обратное преобразование строки битов в действительное значение переменной x_j выполняется по следующей формуле:

$$x_j = a_j + \textit{десятичное\_число}(\textit{строка\_битов}_j) \times \frac{b_j - a_j}{2^{m_j} - 1},$$

где *десятичное_число* (*строка_битов* $_j$) представляет собой десятичное значение, закодированное в бинарной строке *строка_битов* $_j$

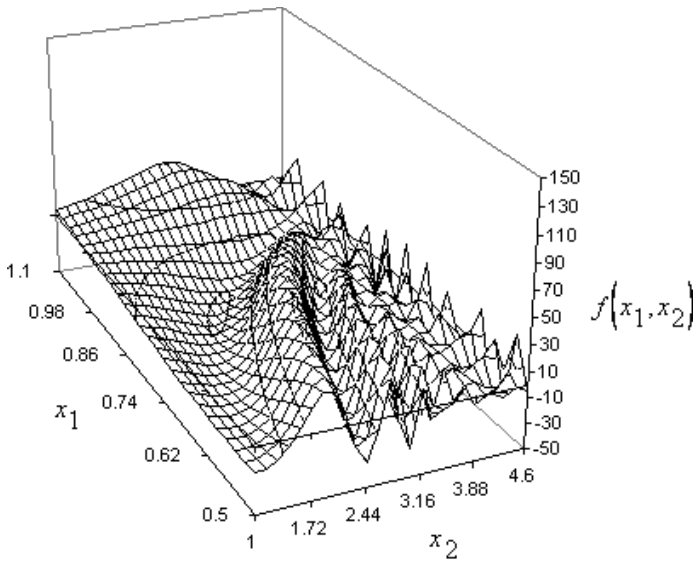


Рис. 1.8. Целевая функция

Предположим, что требуемая точность составляет 5 знаков после запятой. Найдем число битов, необходимых для кодирования переменных x_1 и x_2:

(1.1 - 0.5) $\times$ 100,000 = 60,000

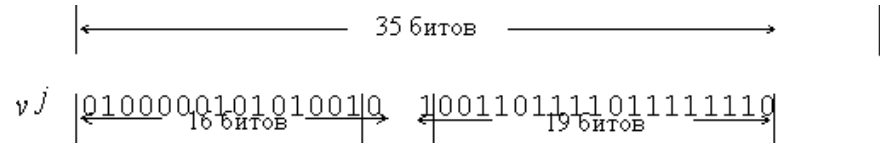
$2^{15} < 60{,}000 \le 2^{16} - 1$, $m_1 = 16$

(4.6 - 1.0) $\times$ 100,000 = 360,000

$2^{18} < 360{,}000 \le 2^{19} - 1$, $m_2 = 19$

$m = m_1 + m_2 = 16 + 19 = 35$.

Суммарная длина хромосомы составляет 35 битов, которые можно представить следующим образом:



Соответствующие значения переменных x_1 и x_2 даны ниже:

	Двоичное число	Десятичное число
x_1	0100000101010010	16722
x_2	1001101111011111110	319230

$$x_1 = 0.5 + 16722 \times \frac{1.1 - 0.6}{2^{16} - 1} = 0.65310,$$

$$x_2 = 1.0 + 319230 \times \frac{4.6 - 1.0}{2^{19} - 1} = 3.19198.$$

Исходная популяция. Исходная популяция генерируется случайно:

v_1 = [01000001010100101001101111011111110]

v_2 = [10001110101110011000000010101001000]

v_3 = [11111000111000001000010101001000110]

v_4 = [01100110110100101101000000010111001]

v_5 = [00000010111101100010001110001101000]

v_6 = [10111110101011011000000010110011001]

v_7 = [00110100010011111000100110011101101]

v_8 = [11001011010100001100010110011001100]

v_9 = [01111110001011101100011101000111101]

v_{10} = [01111101001110101010000010101101010]

Соответствующие значения переменных x_1 и x_2 имеют вид:

v_1 = [x_1 ,x_2] = [0.653097, 3.191983]

v_2 = [x_1 ,x_2] = [0.834511,2.809287]

v_3 = [x_1 ,x_2] = [1.083310,2.874312]

v_4 = [x_1 ,x_2] = [0.740989,3.926276]

v_5 = [x_1 ,x_2] = [0.506940,1.499934]

v_6 = [x_1 ,x_2] = [0.946903,2.809843]

v_7 = [x_1 ,x_2] = [0.622600,2.935225]

v_8 = [x_1 ,x_2] = [0.976521,3.778750]

v_9 = [x_1 ,x_2] = [0.795738,3.802377]

v_{10} = [x_1 ,x_2] = [0.793504,3.259521]

Функция соответствия. Оценка функции соответствия хромосомы выполняется в три шага:

1°. Преобразовать генотип хромосомы в фенотип. В данной задаче это означает преобразование двоичной строчки в соответствующее действительное значение $\boldsymbol{x}^k = \left(x_1^k , x_2^k \right), k = 1,2,...,pop\_size$, где pop_size - число вариантов в исходной популяции.

2°. Вычислить целевую функцию $f\left(\boldsymbol{x}^k \right)$.

3°. Преобразовать целевую функцию в значение функции соответствия. Для решаемой задачи оптимизации функция соответствия эквивалентна целевой функции. $eval\left(\boldsymbol{v}_k \right) = f\left(\boldsymbol{x}^k \right), k = 1,2,...,pop\_size$.

Функция соответствия играет роль среды и оценивает хромосомы по степени их приспособленности к выполнению критерия оптимизации.

Значения функций соответствия вышеприведенных хромосом следующие:

$eval\left(v_1 \right)$ = *f*(0.653097,3.191983) = 20.432394

$eval\left(v_2 \right)$ = *f*(0.834511,2.809287) = -4.133627

$eval\left(v_3 \right)$ = *f*(1.083310,2.874312) = 28.978472

$eval\left(v_4 \right)$ = *f*(0.740989,3.926276) = -2.415740

$eval\left(v_5 \right)$ = *f*(0.506940,1.499934) = -2.496340

$eval(v_6)$ = f(0.946903,2.809843) = -23.503709

$eval(v_7)$ = f(0.622600,2.935225) = -13.878172

$eval(v_8)$ = f(0.976521,3.778750) = -8.996062

$eval(v_9)$ = f(0.795738,3.802377) = 6.982708

$eval(v_{10})$ = f(0.793504,3.259521) = 6.201905

Очевидно, что хромосома v_3 наиболее сильная, а хромосома v_6 наиболее слабая.

Отбор. Наибольшее распространение на практике получил подход, называемый *колесо рулетки* [58] (от англ. roulette wheel). Согласно этому подходу отбор осуществляется на основе некоторой функции распределения, которая строится пропорционально вычисленным функциям соответствия сгенерированных вариантов-хромосом. Колесо рулетки может быть сконструировано следующим образом:

1. Вычисляем значение функции соответствия $eval(v_k)$ для каждой хромосомы v_k:

$$eval(v_k) = f(x^k),\ k = 1,2,...,pop\_size .$$

2. Вычисляем общую функцию соответствия популяции:

$$F = \sum_{k=1}^{pop\_size} \left(eval(v_k) - \min_{j=1,pop\_size} \{ eval(v_j) \} \right)$$

3. Вычисляем вероятность отбора p_k для каждой хромосомы v_k:

$$p_k = \frac{eval(v_k) - \min_{j=1,pop\_size} \{ eval(v_j) \}}{F},\ k = 1,2,...,pop\_size .$$

4. Вычисляем совокупную вероятность q_k для каждой хромосомы v_k:

$$q_k = \sum_{j=1}^{k} p_j ,\ k = 1,2,...,pop\_size .$$

Процесс отбора начинается с вращения колеса pop_size раз; при этом каждый раз выбирается одна хромосома по следующему алгоритму:

1°. Генерируем случайное число r из интервала $[0,1]$.

2°. Если $r \le q_1$, то выбираем первую хромосому v_1; иначе выбираем k -ую хромосому v_k ($2 \le k \le pop\_size$) такую, что $q_{k-1} < r \le q_k$.

Общая функция соответствия F всей популяции равна:

$$F = \sum_{k=1}^{10} \left(eval(v_k) - \min_{j=1,10} \{ eval(v_j) \} \right) = 242.208919 .$$

Вероятность отбора p_k для каждой хромосомы v_k ($k = 1,2,...,10$) равна:

p_1 = 0.181398, p_2 = 0.079973, p_3 = 0.216681,

p_4 = 0.087065, p_5 = 0.086732, p_6 = 0.000000,

p_7 = 0.039741, p_8 = 0.059897, p_9 = 0.125868,

p_{10} = 0.122645 .

Совокупные вероятности q_k для каждой хромосомы v_k ($k = 1, 2, ..., 10$) равны:

q_1 = 0.181398, q_2 = 0.261370, q_3 = 0.478052,

q_4 = 0.565117, q_5 = 0.651849, q_6 = 0.651849,

q_7 = 0.691590, q_8 = 0.751487, q_9 = 0.877355,

q_{10} = 1.000000 .

Теперь мы готовы вращать колесо рулетки 10 раз и каждый раз отобрать одну хромосому для новой популяции. Допустим, что случайная последовательность 10 чисел из интервала $[0, 1]$ имеет вид:

0.301431 0.322062 0.766503 0.881893
0.350871 0.583392 0.177618 0.343242
0.032685 0.197577 .

Первое число r_1 = 0.301431 больше чем q_2 и меньше чем q_3. Это означает, что отбирается хромосома v_3. Второе число r_2 = 0.322062 также больше чем q_2 и меньше чем q_3. Значит опять отбираем хромосому v_3 для новой популяции; и т.д. Наконец, получим новую популяцию, состоящую из таких хромосом:

v'_1 = [11111000111000001000010101001000110] (v_3)

v'_2 = [11111000111000001000010101001000110] (v_3)

v'_3 = [11001011010100001100010110011001100] (v_8)

v'_4 = [01111110001011101100011101000111101] (v_9)

v'_5 = [11111000111000001000010101001000110] (v_3)

v'_6 = [01100110110100101101000000010111001] (v_4)

v'_7 = [01000001010100101001101111011111110] (v_1)

v'_8 = [11111000111000001000010101001000110] (v_3)

v'_9 = [01000001010100101001101111011111110] (v_1)

v'_{10} = [10001110101110011000000010101001000] (v_2)

Скрещивание. Для скрещивания хромосом будем использовать метод с одной точкой обмена. В соответствие с этим методом, случайно выбирается одна точка обмена, относительно которой меняются местами части хромосом-родителей. Для примера рассмотрим скрещивание двух хромосом, для которых была случайно выбрана точка обмена после 17-го гена:

v_1 = [11111000111000001000010101001000110]

v_2 = [10001110101110011000000010101001000]

В результате обмена частей родительских хромосом получаются следующие хромосомы-отпрыски:

$v_1 =$ [1111100011100000 1000000101010010 00]
$v_2 =$ [1000111010111001 1000010100100011 0]

Пусть вероятность скрещивания $p_c = 0.25$, т. е. в среднем 25% хромосом подвергнуться скрещиванию. Таким образом, скрещивание осуществляется по такому алгоритму:

Процедура: Скрещивание.

begin

k :=0 ;

while ($k \le 10$) **do**

r_k := случайное число из интервала [0,1] ;

if ($r_k < 0.25$) **then**

отобрать хромосому v_k как одного из родителей для скрещивания ;

end ;

$k := k + 1$;

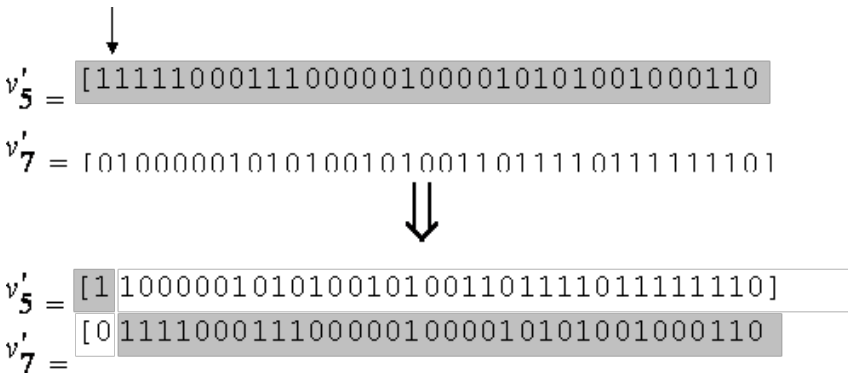
end ;

end.

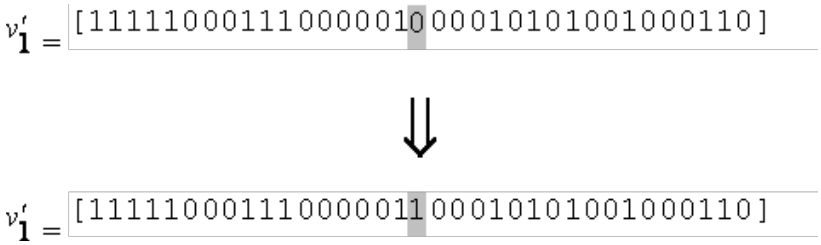
Предположим, что сгенерирована последовательность случайных чисел:

0.625721 0.266823 0.288644 0.295114
0.163274 0.567461 0.085940 0.392865
0.770714 0.548656 .

Это означает, что хромосомы v'_5 и v'_7 выбираются для скрещивания. После этого мы генерируем случайное целое число pos (позиция) из промежутка [1,34] (так как общая длина хромосомы равна 35) и считаем его точкой обмена хромосом или, другими словами, точкой скрещивания. Предположим, что было сгенерировано число pos равное 1, т. е. хромосомы-родители обмениваются частями после первого бита и появятся следующие хромосомы-отпрыски:



Мутация. Мутация состоит в изменении одного или большего числа генов с вероятностью равной коэффициенту мутации. Допустим, что 18-й ген хромосомы v'_1 подвергается мутации. Так как мы имеем дело с бинарными строчками, то мутация заключается в инверсии соответствующего бита:



Зададим коэффициенту мутации значение $p_m = 0.01$, так что в среднем 1% всех битов популяции подвергнуться операции мутации. Число битов во всей популяции составляет $m \times pop\_size = 35 \times 10 = 350$ битов. Поэтому в среднем в каждом поколении мутирует 3.5 бита. Каждый

бит имеет одинаковый шанс подвергнуться мутации. Таким образом, мы должны сгенерировать последовательность случайных чисел r_k $(k = 1, ..., 350)$ из интервала [0,1]. Предположим, что мутируют следующие гены:

Позиция бита в популяции	Номер хромосомы	Позиция бита в хромосоме	Случайное число r_k
111	4	6	0.009857
172	5	32	0.003113
211	7	1	0.000946
347	10	32	0.001282

После мутации получается следующая популяция:

v'_1 = [11111000111000001000010101001000110]

v'_2 = [11111000111000001000010101001000110]

v'_3 = [11001011010100001100010110011001100]

v'_4 = [01111010001011101100011101000111101]

v'_5 = [11000001010100101001101111011110110]

v'_6 = [01100110110100101101000000010111001]

v'_7 = [11111000111000001000010101001000110]

v'_8 = [11111000111000001000010101001000110]

v'_9 = [01000001010100101001101111011111110]

v'_{10} = [10001110101110011000000010101000000] .

Соответствующие десятичные значения переменных x_1 и x_2 и значения функции соответствия имеют вид:

f(1.083310,2.874312)=28.978472

f(1.083310,2.874312)=28.978472

f(0.976521,3.778750)=-8.996062

f(0.786363,3.802377)=9.366723

f(0.953101,3.191928)=-23.229745

f(0.740989,3.926276)=-2.415740

f(1.083310,2.874312)=28.978472

f(1.083310,2.874312)=28.978472

f(0.653097,3.191983)=20.432394

f(0.834511,2.809232)=-4.138564

Таким образом, завершена одна итерация генетического алгоритма. Проделав 1000 итераций, мы получим наилучшую хромосому в 419-м поколении:

v^* = [01000011000100110110010011011101001]

$eval(v^*)$ = *f*(0.657208,2.418399) = 31.313555

x_1^* = 0.657208 x_2^* = 2.418399

$f(x_1^*, x_2^*)$ = 31.313555 .

Пример 1.10. Синтез текста. Этот пример прекрасно демонстрирует мощь генетических алгоритмов. Задача синтеза текста заключается в получении фрагмента известной украинской песни *несе Галя воду*

из случайно сгенерированного списка букв при помощи генетического алгоритма. Так как для каждой позиции текста возможны 32 различные буквы алфавита, а таких позиций в заданном выражении 12, то вероятность получения необходимой строки случайным способом равна

$\left(\frac{1}{31}\right)^{12} = 1.27 \times 10^{-18}$, т. е. практически нуль.

Для кодирования строки букв будем использовать кодовую таблицу символов ASCII (в операционной системе Windows). Тогда строка *несе Галя воду* преобразуется в следующую хромосому:

[237, 229, 241, 229, 227, 224, 235, 255, 226, 238, 228, 243]

Сгенерируем исходную популяцию из 10 случайных фраз:

[232, 239, 225, 242, 227, 238, 255, 227, 186, 238, 236, 239]

[228, 226, 244, 231, 231, 224, 226, 224, 237, 248, 243, 247]

[252, 241, 243, 228, 228, 225, 246, 225, 234, 186, 230, 246]

[249, 227, 252, 249, 244, 245, 236, 229, 248, 252, 224, 226]

[230, 232, 234, 245, 231, 254, 227, 245, 238, 247, 242, 232]

[232, 228, 227, 245, 230, 226, 232, 179, 247, 255, 238, 186]

[232, 248, 231, 235, 248, 179, 235, 186, 224, 255, 252, 242]

[239, 248, 236, 229, 179, 233, 232, 244, 249, 245, 252, 230]

[249, 232, 236, 229, 240, 244, 227, 243, 230, 244, 254, 242]

[248, 230, 231, 252, 245, 239, 242, 254, 252, 255, 240, 255]

Теперь преобразуем эти хромосомы в строки символов и увидим, что получится 10 бессмысленных фраз:

ипбтгоягєомп

двфззаваншуч

ьсуддбцбкєжц

щгьщфхмешьав

жикхзюгхочти

идгхжви_чяоє

ишзлш_лєаяьт

пшме_йифщхьж

щимерфгужфют

шжзьхптюьяря .

Функция соответствия вычисляется как число правильно отгаданных букв. Например, функция соответствия для строки <ипбтгоягєомп> равна 2. В табл. 1.2 приведен список хромосом, которые были наилучшими на каждой из 32-х итераций генетического алгоритма.

Таблица 1.2

Лучшие хромосомы на каждой итерации

Поко-ление	Строка	Функция соответствия	Поколение	Строка	Функция соответствия
1	ипбтгоягєомп	2	17	несечалядгду	9
2	ипбтгоягєомп	2	18	несечалядгду	9
3	ипбтгоягєомп	2	19	несегалядгду	10
4	никхгаваншйт	3	20	несегалядгду	10
5	ндмералядгжц	5	21	несегалядгду	10
6	ндмералядгжц	5	22	несегалядгду	10
7	ндмералядгжц	5	23	несегалядгду	10
8	ндшеьалявгжц	6	24	несегалядоду	11
9	ндшеьалявгжц	6	25	несегалядоду	11
10	ндшеьалявгжц	6	26	несегалядоду	11
11	нисералядгжу	7	27	несегалядоду	11
12	нисералядгжу	7	28	несегалядоду	11
13	нисералядгжу	7	29	несегалядоду	11
14	несеролядгду	8	30	несегалядоду	11
15	несеролядгду	8	31	несегалядоду	11
16	несеролядгду	8	32	несегаляводу	12

В оглавление книги \ К следующему разделу \ К предыдущему разделу