Глава 9

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ

Данная глава посвящена генетическим алгоритмам (ГА), используемым для решения задач оптимизации и моделирования. В разделе 9.1 описаны предпосылки к возникновению эволюционных вычислений и перечислены основные задачи, решаемые с использованием ГА. Общая схема и краткое описание работы ГА представлено в разделе 9.2. Раздел 9.3. посвящен описанию основных этапов и параметров ГА, а в разделе 9.4. приводятся общие рекомендации по настройке значений параметров генетического алгоритма. В разделе 9.5. описан ставший классическим канонический ГА. Раздел 9.6. содержит пример применения и анализа ГА для решения задачи численной оптимизации. Общие замечания по программной реализации генетического алгоритма представлены в разделе 9.7. В разделе 9.8. содержатся варианты заданий для лабораторных работ.

9.1. Введение

Развитие природных систем на протяжении многих веков привлекало внимание ученых. Неоднократно совершались попытки выделить и осмыслить основополагающие принципы и механизмы, лежащие в основе изменений, происходящих в живой природе. Предлагалось множество различных концепций, пока в 1858 году Чарльз Дарвин не опубликовал свою знаменитую работу «Происхождение видов», в которой были провозглашены принципы наследственности, изменчивости и естественного отбора. Однако на протяжении почти 100 последующих лет оставались неясными механизмы, отвечающие за наследственность и изменчивость организмов. В 1944 году Эйвери, Маклеод и Маккарти опубликовали результаты своих исследований, доказывавших, что за наследственные процессы ответственна «кислота дезоксирибозного типа». Это открытие послужило толчком к многочисленным исследованиям во всем мире, и 27 апреля 1953 года в журнале «Nature» вышла статья Уотсона и Крика, где была описана модель двухцепочечной спирали ДНК.

Знание эволюционных принципов и генетических основ наследственности позволило разработать как модели молекулярной эволюции [Редько, 2003], описывающие динамику изменения молекулярных последовательностей, так и макроэволюционные модели, используемые в экологии, истории и социологии для исследования экосистем и сообществ организмов [Редько, 2003, Бурцев, 2005].

Эволюционные принципы используются не только для моделирования, но и для решения прикладных задач оптимизации. Множество алгоритмов и методов, использующие для поиска решения эволюционный подход, объеди-

няют под общим названием **эволюционные вычисления (ЭВ)** или **эволюционные алгоритмы (ЭА)** [Beyer, Schwefel, Wegener, 2002]. Выделяют следующие виды ЭА:

- генетический алгоритм [Holland, 1975, Емельянов и др., 2003];
- эволюционное программирование [Фогель и др., 1969];
- эволюционные стратегии [Rechenberg, 1973, Schwefel, 1977];
- генетическое программирование [Koza, 1992].

В данной главе будет рассмотрен генетический алгоритм (ГА) как один из самых распространенных эволюционных алгоритмов. Краткое описание видов ЭА приведено в [Whitley, 2002].

Круг задач, решаемых с помощью ГА очень широк. Ниже перечислены некоторые задачи, для решения которых использовался ГА [Heitkotter, Beasly, 2001]:

- задачи численной оптимизации;
- задачи о кратчайшем пути;
- задачи компоновки;
- составление расписаний;
- аппроксимация функций;
- отбор (фильтрация) данных;
- настройка и обучение искусственной нейронной сети;
- искусственная жизнь;
- биоинформатика;
- игровые стратегии;
- нелинейная фильтрация;
- развивающиеся агенты/машины.

9.2. Генетический алгоритм

Идея генетических алгоритмов предложена Джоном Холландом в 60-х годах, а результаты первых исследований обобщены в его монографии «Адаптация в природных и искусственных системах» [Holland, 1975], а также в диссертации его аспиранта Кеннета Де Йонга [De Jong, 1975].

Как уже говорилось выше, ГА используют для работы эволюционные принципы наследственности, изменчивости и естественного отбора. Общая схема ГА представлена на рис. 9.1.

Генетический алгоритм работает с *популяцией особей*, в *хромосоме (генотип)* каждой из которых закодировано возможное решение задачи (*фенотип*). В начале работы алгоритма популяция формируется случайным образом (блок «**Формирование начальной популяции**» на рис. 9.1). Для того чтобы оценить качество закодированных решений используют *функцию приспособленности*, которая необходима для вычисления *приспособленности* каждой особи (блок «**Оценивание популяции**»). По результатам оценивания особей наиболее приспособленные из них выбираются (блок «**Селекция**») для скрещивания. В результате *скрещивания* выбранных особей посредством применения генетического оператора *кроссовера* создается *потомство*, гене-

тическая информация которого формируется в результате обмена хромосомной информацией между родительскими особями (блок «**Скрещивание**»). Созданные потомки формируют новую популяцию, причем часть потомков *мутирует* (используется генетический оператор *мутации*), что выражается в случайном изменении их генотипов (блок «**Мутация**»). Этап, включающий последовательность «Оценивание популяции» – «Селекция» – «Скрещивание» – «Мутация», называется *поколением*. Эволюция популяции состоит из последовательности таких поколений.

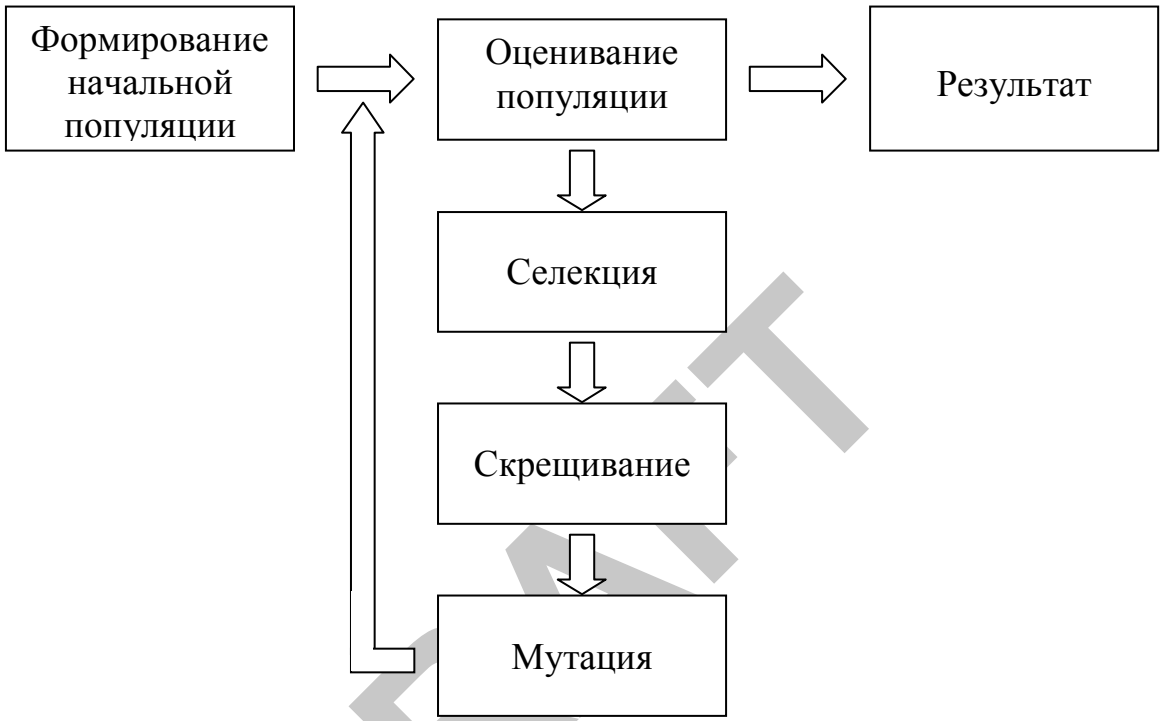

Рис. 9.1 Общая схема генетического алгоритма

Длительность эволюции может определяться следующими факторами:
- нахождение решения в результате эволюционного поиска;
- ограниченность количества поколений;
- вырождение популяции, когда степень разнородности хромосом в популяции становится меньше допустимого значения.

9.3. Параметры и этапы генетического алгоритма

9.3.1. Кодирование информации и формирование начальной популяции

Выбор способа кодирования является одним из важнейших этапов при использовании эволюционных алгоритмов. В частности, должно выполняться следующее условие:

- должна быть возможность закодировать (с допустимой погрешностью) в хромосоме любую точку из пространства поиска.

Невыполнение этого условия может привести как к увеличению времени эволюционного поиска, так и к невозможности найти решение поставленной задачи.

Как правило, в хромосоме кодируются численные параметры решения. Для этого возможно использование целочисленного и вещественного кодирования.

Целочисленное кодирование. В каноническом генетическом алгоритме хромосома представляет собой битовую строку, в которой закодированы параметры решения поставленной задачи. На рис. 9.2 показан пример кодирования 4-х 10-разрядных параметров в 40 разрядной хромосоме. Как правило, считают, что каждому параметру соответствует свой *ген*. Таким образом, можно также сказать, что хромосома на рис. 9.2 состоит из 4-х 10-разрядных генов.

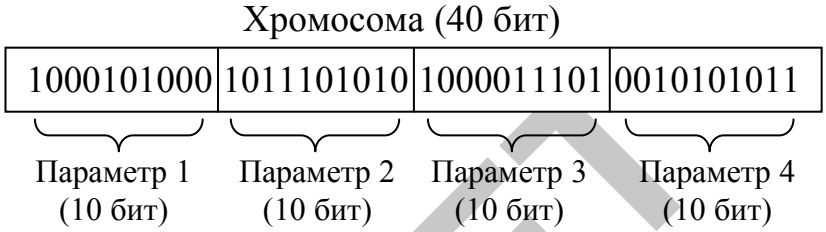

Рис. 9.2. Пример целочисленного кодирования

Несмотря на то, что каждый параметр закодирован в хромосоме целым числом, ему могут быть поставлены в соответствие и вещественные числа. Ниже представлен один из вариантов прямого и обратного преобразования «целочисленный ген → вещественное число».

Если известен диапазон, в пределах которого лежит значение параметра, то этот диапазон разбивают на 2^m отрезков, где m – разрядность гена, и каждому отрезку соответствует определенное значение гена. При этом для перевода значений из закодированного значения в дробные применяют следующие формулы:

$$r = \frac{g(x_{\max} - x_{\min})}{2^m - 1} + x_{\min},$$

$$g = \frac{(r - x_{\min})(2^m - 1)}{(x_{\max} - x_{\min})},$$

где r – вещественное (декодированное) значение параметра, g – целочисленное (закодированное) значение параметра, $x_{\max}$ и $x_{\min}$ – соответственно максимальное и минимальное допустимое значение декодированного параметра.

Например, если искомое значение параметра лежит в промежутке [1;2], и каждый ген кодируется 16 разрядами, то, если содержимое гена равно $ABCD_{16} = 43981_{10}$, то соответствующее дробное значение равно

$$r = 43981 * (2 - 1)/(2^{16} - 1) + 1 = 0{,}6711 + 1 = 1{,}6711.$$

Если же декодированное значение равно 1,3275, то соответствующий ген после обратного преобразования будет содержать (с округлением в меньшую сторону):

$$g = (1{,}3275 - 1)(2^{16} - 1) / (2 - 1) = 0{,}3275 * 65535 = 21462{,}7125 \approx 21462_{10} = 0101\ 0011\ 1101\ 0110_2$$

Вещественное кодирование. Часто бывает удобнее кодировать в гене не целое число, а вещественное. Это позволяет избавиться от операций кодирования/декодирования, используемых в целочисленном кодировании, а также увеличить точность найденного решения. Пример вещественного кодирования представлен на рис. 9.3.

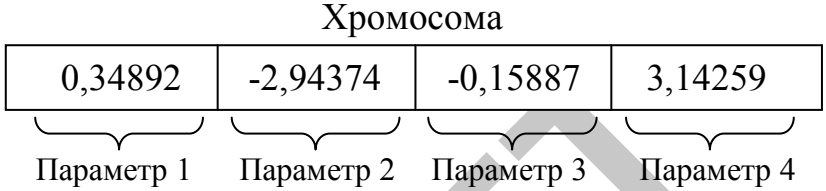

Рис. 9.3. Пример вещественного кодирования

Формирование начальной популяции. Как правило, начальная популяция формируется случайным образом. При этом гены инициализируются случайными значениями. Пример подобной инициализации на псевдоязыке представлен ниже.

```
i = 0;
ПОКА  (i < РАЗМЕР_ПОПУЛЯЦИИ) {
    j = 0;
    ПОКА (j < ЧИСЛО_ГЕНОВ) {
        ОСОБЬ[i].ГЕН[j] = СЛУЧАЙНАЯ_ВЕЛИЧИНА;
        j = j+1;
    }
    i = i+1;
}
```

9.3.2. Оценивание популяции

Оценивание популяции необходимо для того, чтобы выявить в более приспособленных и менее приспособленных особей. Для подсчета приспособленности каждой особи используется функция приспособленности (*целевая функция*)

$f_i = f(\mathbf{G}_i)$,

где $\mathbf{G}_i = \{ g_{ik} \mid k = 1,2,...,N \}$ – хромосома i-й особи, g_{ik} – значение k-го гена i-й особи, N – количество генов в хромосоме. В случае использования целочисленного кодирования (см. предыдущий раздел) для вычисления значения функции приспособленности часто бывает необходимо преобразовать зако-

дированные в хромосоме целочисленные значения к вещественным числам. Другими словами:

$f(\mathbf{G}_i) = f(\mathbf{X}_i)$,

где $\mathbf{X}_i = \{ x_{ik} \mid k = 1,2,...,N \}$ – вектор вещественных чисел, соответствующих генам i-й хромосомы.

Как правило, использование эволюционного алгоритма подразумевает решение задачи максимизации (минимизации) целевой функции, когда необходимо найти такие значения параметров функции f, при которых значение функции максимально (минимально). В соответствии с этим, если решается задача минимизации, то если $f(\mathbf{G}_i) < f(\mathbf{G}_j)$, то считают, что i-я особь лучше (приспособленнее) j-й особи. В случае задачи максимизации, наоборот, если $f(\mathbf{G}_i) > f(\mathbf{G}_j)$, то i-я особь приспособленнее, чем j-я особь.

9.3.3. Селекция

Селекция (*отбор*) необходима, чтобы выбрать более приспособленных особей для скрещивания. Существует множество вариантов селекции, опишем наиболее известные из них.

Рулеточная селекция. В данном варианте селекции вероятность i-й особь принять участие в скрещивании p_i пропорциональна значению ее приспособленности f_i и равна

$$p_i = \frac{f_i}{\sum_j f_j},$$

Процесс отбора особей для скрещивания напоминает игру в «рулетку». Рулеточный круг делится на сектора, причем площадь i-го сектора пропорциональна значению p_i. После этого n раз «вращается» рулетка, где n – размер популяции, и по сектору, на котором останавливается рулетка, определяется особь, выбранная для скрещивания.

Селекция усечением. При отборе усечением после вычисления значений приспособленности для скрещивания выбираются ln лучших особей, где l – «порог отсечения», $0 < l < 1$, n – размер популяции. Чем меньше значение k, тем сильнее давление селекции, т.е. меньше шансы на выживание у плохо приспособленных особей. Как правило, выбирают l в интервале от 0,3 до 0,7.

Турнирный отбор. В случае использования турнирного отбора для скрещивания, как и при рулеточной селекции, отбираются n особей. Для этого из популяции случайно выбираются t особей, и самая приспособленная из них допускается к скрещиванию. Говорят, что формируется турнир из t особей, t – размер турнира. Эта операция повторяется n раз. Чем больше значение t, тем больше давление селекции. Вариант турнирного отбора, когда $t = 2$, называют бинарным турниром. Типичные значения размера турнира $t = 2, 3, 4, 5$.

9.3.4. Скрещивание и формирование нового поколения

Отобранные в результате селекции особи (называемые *родительскими*) скрещиваются и дают потомство. Хромосомы потомков формируются в про-

цессе обмена генетической информацией (с применением оператора *кроссовера*) между родительскими особями. Созданные таким образом потомки составляют популяцию следующего поколения. Ниже будут описаны основные скрещивания для целочисленного и вещественного кодирования. Будем рассматривать случай, когда из множества родительских особей случайным образом выбираются 2 особи и скрещиваются с вероятностью P_C , в результате чего создаются 2 потомка. Этот процесс повторяется до тех пор, пока не будет создано n потомков. Вероятность скрещивания P_C является одним из ключевых параметров генетического алгоритма и в большинстве случаев ее значение находится в диапазоне от 0,6 до 1. Процесс скрещивания на псевдоязыке выглядит следующим образом (предполагается, что размер подпопуляции родительских особей равен размеру популяции, RANDOM – случайное число из диапазона [0; 1]):

```
k = 0;
ПОКА (k < РАЗМЕР_ПОПУЛЯЦИИ) {
     i = RANDOM * РАЗМЕР_ПОПУЛЯЦИИ;
     j = RANDOM * РАЗМЕР_ПОПУЛЯЦИИ;
     ЕСЛИ (PC > RANDOM) {
          СКРЕЩИВАНИЕ (РОДИТЕЛЬ[i], РОДИТЕЛЬ[j],
                       ПОТОМОК[k], ПОТОМОК[k+1]);
          k = k+2;
     }
}
```

Целочисленное кодирование. Для целочисленного кодирования часто используются 1-точечный, 2-точечный и однородный операторы кроссовера.

1-точечный кроссовер работает аналогично операции перекреста для хромосом при скрещивании биологических организмов. Для этого выбирается произвольная точка разрыва и для создания потомков производится обмен частями родительских хромосом. Иллюстративный пример работы 1-точечного кроссовера представлен на рис. 9.4.

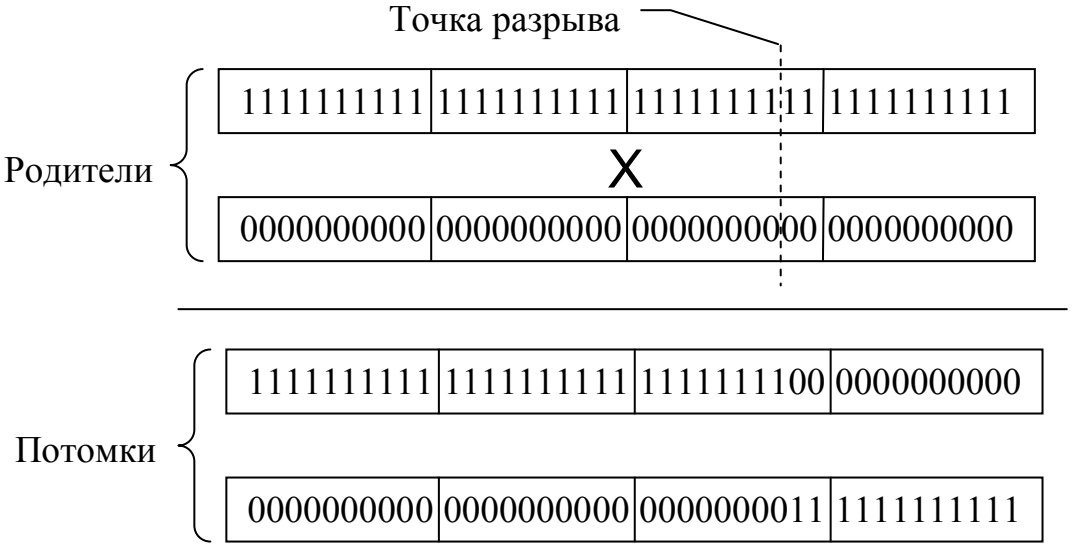

Рис. 9.4. Пример работы 1-точечного кроссовера

Для оператора *2-точечного кроссовера* выбираются 2 случайные точки разрыва, после чего для создания потомков родительские хромосомы обмениваются участками, лежащими между точками разрыва.

При использовании *однородного оператора кроссовера* разряды родительских хромосом наследуется независимо друг от друга. Для этого определяют вероятность p_0, что i-й разряд хромосомы 1-го родителя попадет к первому потомку, а 2-го родителя – ко второму потомку. Вероятность противоположного события равна $(1 - p_0)$. В большинстве случаев вероятность обоих событий одинакова, т.е. $p_0 = 0{,}5$.

Вещественное кодирование. Для вещественного кодирования рассмотрим 2-точечный, арифметический и *BLX-*α операторы кроссовера.

2-точечный кроссовер для вещественного кодирования, в целом, аналогичен 2-точечному кроссоверу для целочисленного кодирования. Различие заключается в том, что точка разрыва не может быть выбрана «внутри» гена, а должна попасть между генами (рис. 9.5).

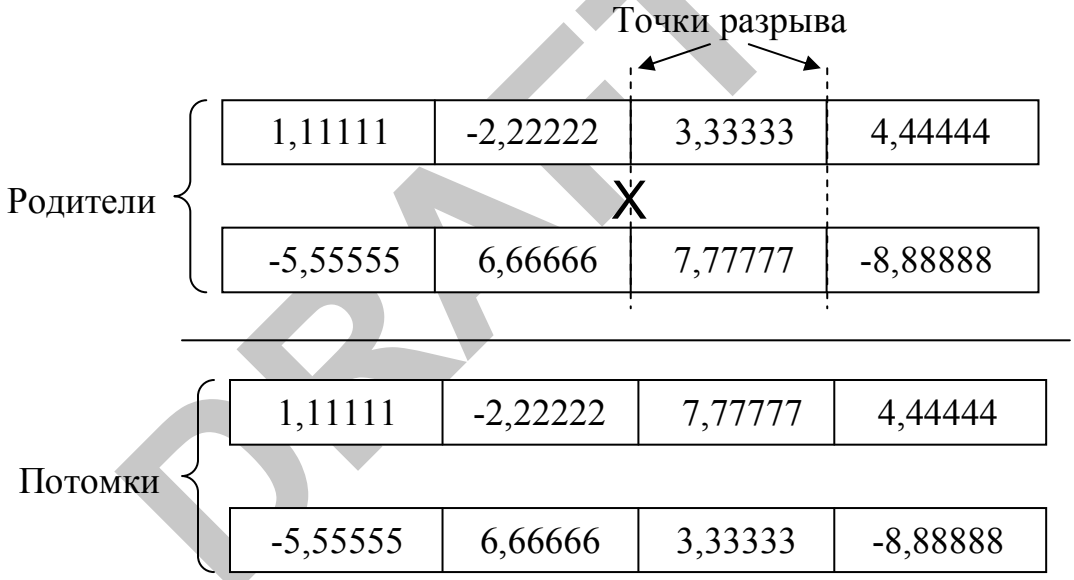

Рис. 9.5. Пример работы 2-хточечного кроссовера для вещественного кодирования

При использовании арифметического и *BLX-*α операторов обмен информацией между родительскими особями и потомками производится с учетом значений генов родителей.

Обозначим $g_k^{(1)}$ и $g_k^{(2)}$ – k-е гены родительских особей, $1 \le k \le N$, N – количество генов в хромосоме. Пусть также $h_k^{(1)}$ и $h_k^{(2)}$ – k-е гены потомков. Тогда для арифметического кроссовера:

$$h_k^{(1)} = \lambda g_k^{(1)} + (1-\lambda) g_k^{(2)},$$

$$h_k^{(2)} = \lambda g_k^{(2)} + (1-\lambda) g_k^{(1)},$$

где $0 \le \lambda \le 1$.

Если используется *BLX-*α кроссовер, то значение *k*-го гена потомка выбирается случайным образом (равномерное распределение) из интервала [c_{min} – $\alpha\Delta_k$, c_{max} + $\alpha\Delta_k$], где α – константа,

$$c_{\min} = \min\{g_k^{(1)}, g_k^{(2)}\},$$

$$c_{\max} = \max\{g_k^{(1)}, g_k^{(2)}\},$$

$$\Delta_k = c_{\max} - c_{\min}.$$

Изображение интервала, используемого для *BLX-*α кроссовера показано на рис. 9.6.

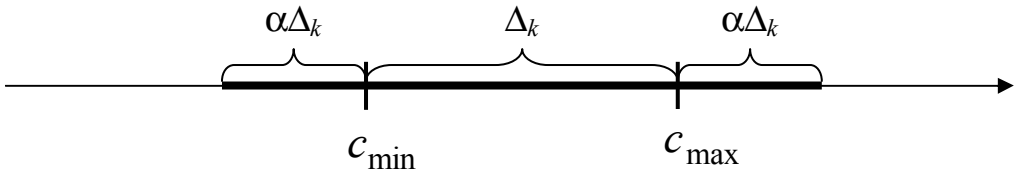

Рис. 9.6. Интервал для *BLX-*α кроссовера

Разрушающая способность кроссовера. Операторы кроссовера характеризуются *способностью к разрушению* (*dispurtion*) родительских хромосом. Кроссовер для целочисленного кодирования считается более разрушительным, если в результате его применения расстояние по Хэммингу между получившимися хромосомами потомков и хромосомами родителей велико. Другими словами, способность целочисленного кроссовера к разрушению зависит от того, насколько сильно он «перемешивает» (рекомбинирует) содержимое родительских хромосом. Так, 1-точечный кроссовер считается слаборазрушающим, а однородный кроссовер в большинстве случаев является сильноразрушающим оператором. Соответственно, 2-точечный кроссовер по разрушающей способности занимает промежуточную позицию по отношению к 1-точечному и однородному кроссоверам.

Для кроссовера для вещественного кодирования способность к разрушению определяется тем, насколько велико расстояние в пространстве поиска между точками, соответствующими хромосомам родителей и потомков. Таким образом, разрушающий эффект 2-точечного кроссовера зависит от содержимого родительских хромосом. Разрушающая способность арифметического кроссовера зависит от значения параметра λ, например, при $\lambda \to 1$ и $\lambda \to 0$, способность к разрушению будет низкой. Для *BLX-*α кроссовера разрушающая способность зависит как от значения α, так и от разности значений соответствующих генов родительских особей.

Отметим, что одновременно со способностью к разрушению говорят также о способности к созданию (*creation, construction*) кроссовером новых особей. Тем самым подчеркивается, что, разрушая хромосомы родительских особей, кроссовер может создать совершенно новые хромосомы, не встречавшиеся ранее в процессе эволюционного поиска.

Формирование нового поколения. Как уже упоминалось выше, в результате скрещивания создаются потомки, которые формируют популяцию следующего поколения.

Отметим, что обновленная таким образом популяция не обязательно должна включать одних только особей-потомков. Если доля обновляемых особей равна T, то в новое поколение попадает Tn потомков, n – размер популяции, а $(1 - T)n$ особей в новой популяции являются наиболее приспособленными родительскими особями (так называемые *элитные особи*). Параметр T называют *разрыв поколений* (*generation gap*) [De Jong, 1975]. Использование элитных особей позволяет увеличить скорость сходимости генетического алгоритма.

9.3.5. Мутация

Оператор мутации используется для внесения случайных изменений в хромосомы особей. Это позволяет «выбираться» из локальных экстремумов и, тем самым, эффективнее исследовать пространство поиска. Так же, как и для оператора кроссовера, существует вероятность применения мутации P_M.

Рассмотрим различные операторы мутации в зависимости от способа представления генетической информации.

Целочисленное кодирование. Опишем оператор битовой мутации. В случае целочисленного кодирования мутация изменяет отдельные разряды в хромосоме. Для этого каждый разряд инвертируется с вероятностью P_M. Ниже приведен пример мутации на псевдоязыке:

```
ДЛЯ КАЖДОЙ k ОСОБИ В ПОПУЛЯЦИИ {
    ДЛЯ КАЖДОГО i РАЗРЯДА В ХРОМОСОМЕ k {
        ЕСЛИ (Pм > RANDOM) {
            БИТОВАЯ_МУТАЦИЯ (ОСОБЬ[k], i);
        }
    }
}
```

В силу того, что применение мутации разыгрывается столько раз, сколько разрядов содержится в хромосоме, значение P_M выбирают небольшим, чтобы сильно не разрушать найденные хорошие хромосомы. Один из типичных вариантов $P_M = L^{-1}$, где L – длина хромосомы в битах, в этом случае каждая хромосома мутирует в среднем один раз.

Вещественное кодирование. Оператор мутации для вещественного кодирования изменяет содержимое каждого гена с вероятностью P_M. При этом величина изменения выбирается случайно в некотором диапазоне $[-\xi; +\xi]$, например, [–0,5; 0,5], и может иметь как равномерное, так и любое другое распределение, к примеру, нормальное с $m_x = 0$, $\sigma_x = 0{,}5$. Таким образом, пример мутации для вещественного кодирования на псевдоязыке выглядит следующим образом (RND – случайное число, распределенное по заранее определенному закону):

```
ДЛЯ КАЖДОЙ k ОСОБИ В ПОПУЛЯЦИИ {
    ДЛЯ КАЖДОГО i ГЕНА В ХРОМОСОМЕ k {
        ЕСЛИ (PМ > RANDOM) {
            ОСОБЬ[k].ГЕН[i] = ОСОБЬ[k].ГЕН[i] + RND;
        }
    }
}
```

Для того чтобы избежать сильных изменений содержимого хромосомы в результате мутации значение вероятности P_M выбирается небольшим. Например, $P_M = N^{-1}$, где N – количество генов в хромосоме.

9.4. Настройка параметров генетического алгоритма

Результат работы генетического алгоритма сильно зависит от того, каким образом настроены его параметры. Основными параметрами ГА являются:

- длительность эволюции (количество поколений);
- размер популяции;
- интенсивность (давление) селекции;
- разновидность оператора кроссовера;
- вероятность кроссовера P_C;
- разновидность оператора мутации;
- вероятность мутации P_M;
- величина разрыва поколений T.

Отметим, что вышеприведенный список может быть легко расширен, но перечисленные параметры присутствуют практически в любой реализации ГА. Различные параметры влияют на разные аспекты эволюционного поиска, среди которых можно выделить два наиболее общих:

1. Исследование пространства поиска (exploration).
2. Использование найденных «хороших» решений (exploitation).

Первый аспект отвечает за способности ГА к эффективному поиску решения и характеризует способности алгоритма избегать локальных экстремумов. Второй аспект важен для постепенного улучшения имеющихся результатов от поколения к поколению на основе уже найденных «промежуточных» решений. Пренебрежение исследовательскими способностями приводит к существенному увеличению времени работы ГА и ухудшению результатов из-за «застревания» алгоритма в локальных экстремумах. В итоге становится возможной *преждевременная сходимость* генетического алгоритма (также говорят о *вырождении популяции*), когда решение еще не найдено, но в популяции практически все особи становятся одинаковыми и долгое время (порядка нескольких десятков и сотен поколений) не наблюдается улучшения приспособленности.

Игнорирование найденных решений может привести к тому, что работа ГА будет напоминать случайный поиск, что также отрицательно сказывается

на эффективности поиска и качестве получаемых решений.

Основная цель в настройке параметров ГА и, одновременно, необходимое условие для стабильного получения хороших результатов работы алгоритма – это достижение **баланса между исследованием пространства поиска и использованием найденных решений**.

Взаимосвязь между параметрами генетического алгоритма, а также их влияние на эволюционный процесс носит сложный характер. На рис. 9.7 схематично изображено влияние изменения некоторых параметров ГА на характеристики эволюционного поиска.

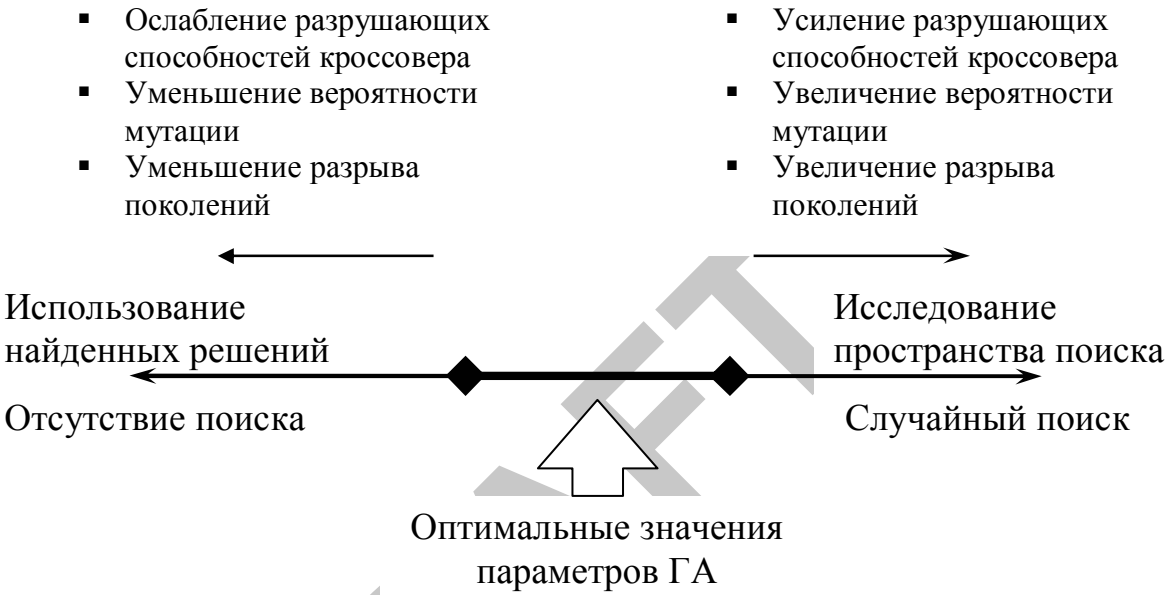

Рис. 9.7. Влияние параметров ГА на характеристики эволюционного поиска

Неправильная настройка параметров может стать причиной различных проблем в работе ГА. Краткий список таких проблем и возможные пути их исправления приведены в табл. 9.1.

Табл. 9.1. Проблемы в работе ГА и возможные пути их исправления

Проблема	**Возможные способы исправления**
1. Плохая приспособленность решений	1. Увеличение числа поколений эволюционного поиска. 2. Увеличение численности популяции. 3. Изменение критерия оценки особей. 4. Исправление способа формирования родительских пар для скрещивания. 5. Исправление стратегии скрещивания и формирования нового поколения.
2. Преждевременная сходимость (вырождение	1. Изменение стратегии выбора родительских пар для скрещивания.

популяции)	2. Отслеживание появления в популяции идентичных особей и их удаление. 3. Использование сильно разрушающего оператора кроссовера. 4. Увеличение вероятности мутации.
3. Низкая «стабильность» эволюции популяции (значительные скачки значений приспособленности от поколения к поколению).	1. Применение “элитизма” (уменьшение разрыва поколений). 2. Уменьшение вероятности мутации. 3. Использование кроссовера со слабой разрушающей способностью.
4. Преобладание удовлетворительных результатов над хорошими.	1. Изменение стратегии выбора родительских пар для скрещивания. 2. Изменение операторов скрещивания и/или мутации. 3. Распараллеливание поиска. Инициализация нескольких независимых популяций, которые развиваются независимо и, время от времени, обмениваются особями.

9.5. Канонический генетический алгоритм.

Канонический генетический алгоритм разработан Джоном Холландом и описан в его книге «Адаптация в естественных и искусственных системах» [Holland, 1975]. Канонического ГА имеет следующие характеристики:

- целочисленное кодирование;
- все хромосомы в популяции имеют одинаковую длину;
- постоянный размер популяции;
- рулеточная селекция;
- одноточечный кроссовер;
- битовая мутация;
- новое поколение формируется только из особей-потомков (разрыв поколений $T = 1$).

9.6. Пример работы и анализа генетического алгоритма

При использовании генетического алгоритма для решения задачи оптимизации необходимо:

1. Определить количество и тип переменных задачи, которые необходимо закодировать в хромосоме.
2. Определить критерий оценки особей, задав функцию приспособленности (целевую функцию).
3. Выбрать способ кодирования и его параметры.

4. Определение параметров ГА (размер популяции, тип селекции, генетические операторы и их вероятности, величина разрыва поколений).

Отметим, что параметры ГА, определяемые в пункте 4 (а также, иногда, в пунктах 2 и 3), часто определяются методом проб и ошибок, на основе анализа получаемых результатов. Отметим, что для анализа результатов работы ГА необходимо произвести несколько запусков алгоритма. Описанная общая схема решения задачи с использованием ГА показана на рис. 9.8.

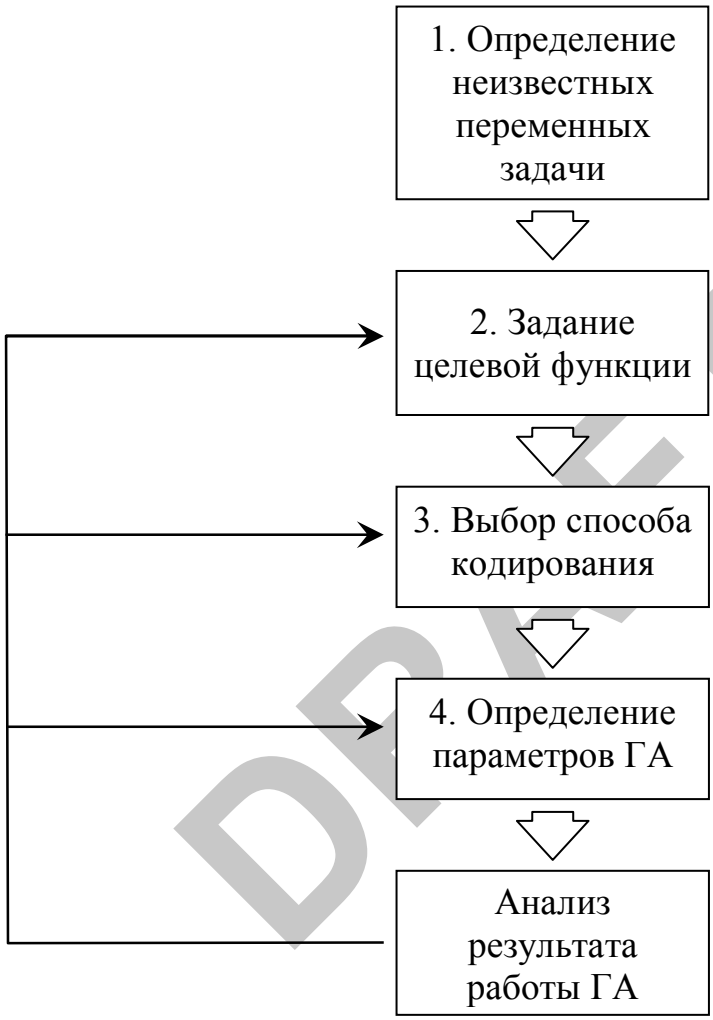

Рис. 9.8. Общая схема решения задачи с использованием ГА

Рассмотрим пример использования ГА для решения задачи минимизации следующей функции (сферическая функция):

$$z = \sum_{i=1}^{n} x_i^2, n = 10, x_i \in [-5{,}12;\ 5{,}12]. \qquad (9.1)$$

Параметр n задает количество переменных функции z. Необходимо найти такие значения переменных x_i, при которых функция z принимает наименьшее значение. Будем использовать общую схему решения (рис. 9.5):

1. Определение неизвестных переменных задачи. По условию поставленной задачи необходимо найти значения переменных x_i, минимизирующие

значение функции z, поэтому в хромосоме будем кодировать значения x_i. Таким образом, каждый i-й ген хромосомы будет соответствовать i-й переменной функции z.

2. Задание функции приспособленности. Будем определять приспособленность особи в зависимости от значения, которое принимает функция z при подстановке в нее вектора параметров, соответствующих генотипу этой особи. Поскольку рассматривается задача минимизации функции z, то будем также считать, что чем меньше значение z, тем приспособленнее особь. Приспособленность i-й особи f_i будем определять по следующей формуле:

$$f_i = \exp(-z_i),$$

где z_i – значение функции z в точке, соответствующей i-й особи.

3. Выбор способа кодирования. В качестве способа представления генетической информации рассмотрим целочисленное кодирование с 10-разрядными генами. Учитывая, что содержимое каждого гена может принимать одно из $2^{10} = 1024$ значений в диапазоне от [-5,12; 5,12], получим, что переменные x_i кодируются в хромосоме с точностью до 0,01.

4. Определение параметров ГА. Для решения задачи рассмотрим популяцию из 20 особей. Для отбора особей для скрещивания будем использовать рулеточную селекцию. В качестве генетических операторов будем использовать 1-точечный кроссовер и битовую мутацию. Вероятности применения операторов скрещивания и мутации установим равными 0,7 и 0,1 соответственно. Новое поколение будем формировать только из особей-потомков, т.е. величина разрыва поколений T равна 1.

Результат работы генетического алгоритма с выбранными параметрами представлен на рис. 9.9. Показаны зависимости изменения среднего $\langle z \rangle$ и наименьшего $z_{\min}$ в популяции значения функции z от номера поколения t. Данные усреднены по 50 независимым запускам.

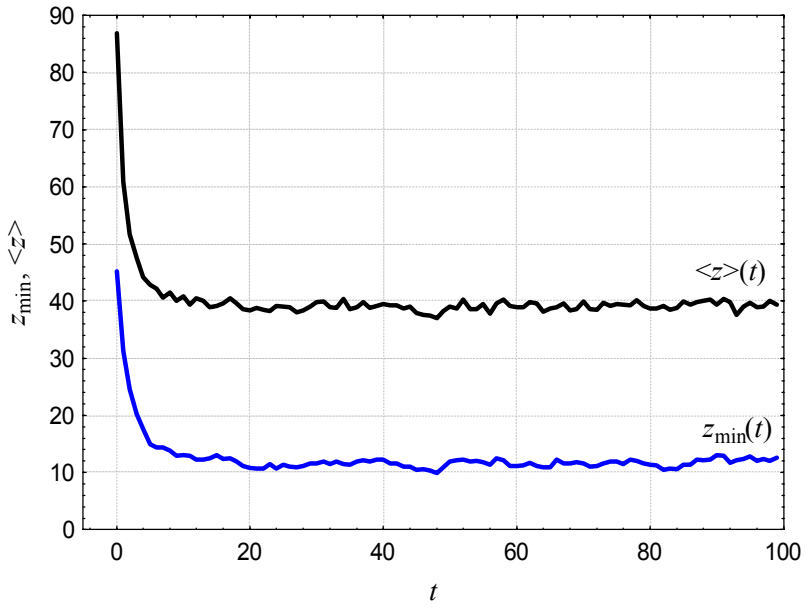

Рис. 9.9. Изменение $z_{\min}(t)$ и $\langle z \rangle(t)$. Популяция из 20 особей, рулеточная селекция, одноточечный кроссовер (P_C = 0,7), битовая мутация (P_M = 0,1).

По данным рис. 9.9 видно, что после 20 поколения значение $z_{\min}$ колеблется в достаточно большом диапазоне. Из этого следует, что потери хороших особей в результате мутации велики, и следует уменьшить вероятность мутации. Установим значение этого параметра равным $L^{-1} = 0{,}01$, где L – длина хромосомы в битах, в данном случае $L = 100$. Результаты работы ГА с измененным значением вероятности мутации показаны на рис. 9.10.

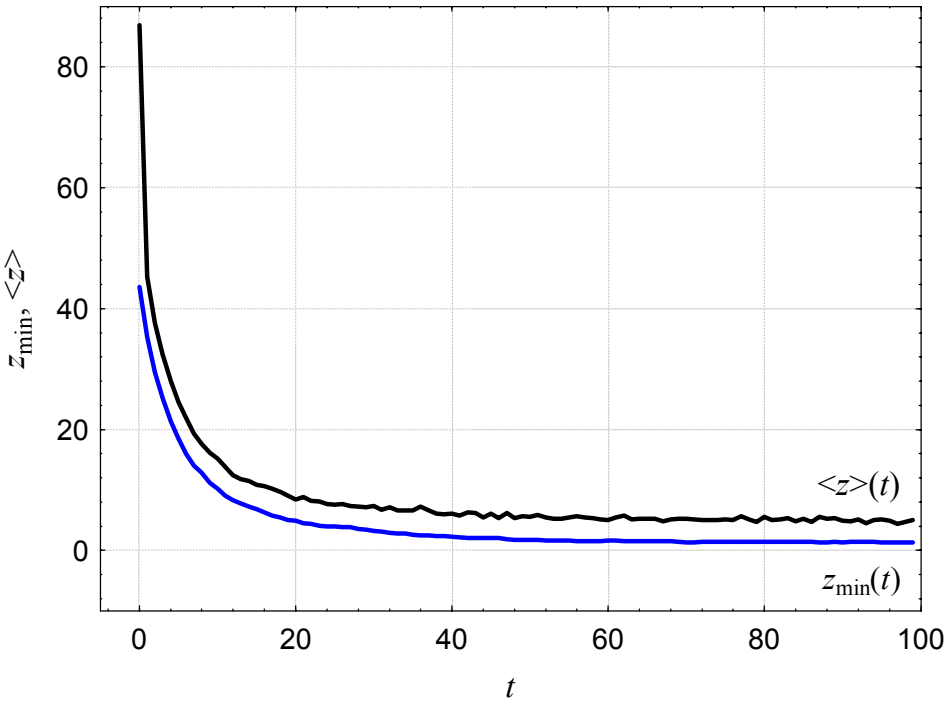

Рис. 9.10. Изменение $z_{\min}(t)$ и $<z>(t)$. Популяция из 20 особей, рулеточная селекция, одноточечный кроссовер ($P_C = 0{,}7$), битовая мутация ($P_M = 0{,}01$).

Из рис. 9.10 видно, что уменьшение вероятности мутации улучшило результат работы ГА. Также отметим, что в теперь эволюционный процесс стабилизировался значительно позднее, примерно после 60 поколения. Минимальное значение функции z, достигнутое за отрезок из первых 100 поколений, равно ~1,27057. Чтобы улучшить результат, изменим используемую рулеточную селекцию на отбор усечением с порогом $l = 0{,}5$, который характеризуется большим давлением селекции. Результат представлен на рис. 9.11.

Увеличение давления селекции привело к ускорению эволюционного поиска за счет удаления из популяции особей со средней и плохой приспособленностью. В результате стабилизация наступила после 40 поколения, а минимальное полученное значение функции z равно 0,00054. Очевидно, что наименьшее значение функции z достигается в точке $x_i = 0$, $i = 1,2,\ldots,10$ и равно 0. В случае поиска минимума функции z с точностью 0,001, для ГА с параметрами, соответствующими графикам на рис. 9.11, решение было найдено в 49 запусках из 50. При этом в среднем было использовано1279,64 вычислений целевой функции. Чтобы повысить стабильность результатов увеличим размер популяции до 50 особей. Полученные кривые $z_{\min}(t)$ и $<z>(t)$

изображены на рис. 9.12. Во всех 50 запусках найден минимум функции z с точностью 0,001. Среднее количество вычислений целевой функции, использованное для нахождения решения, равно 1904,26.

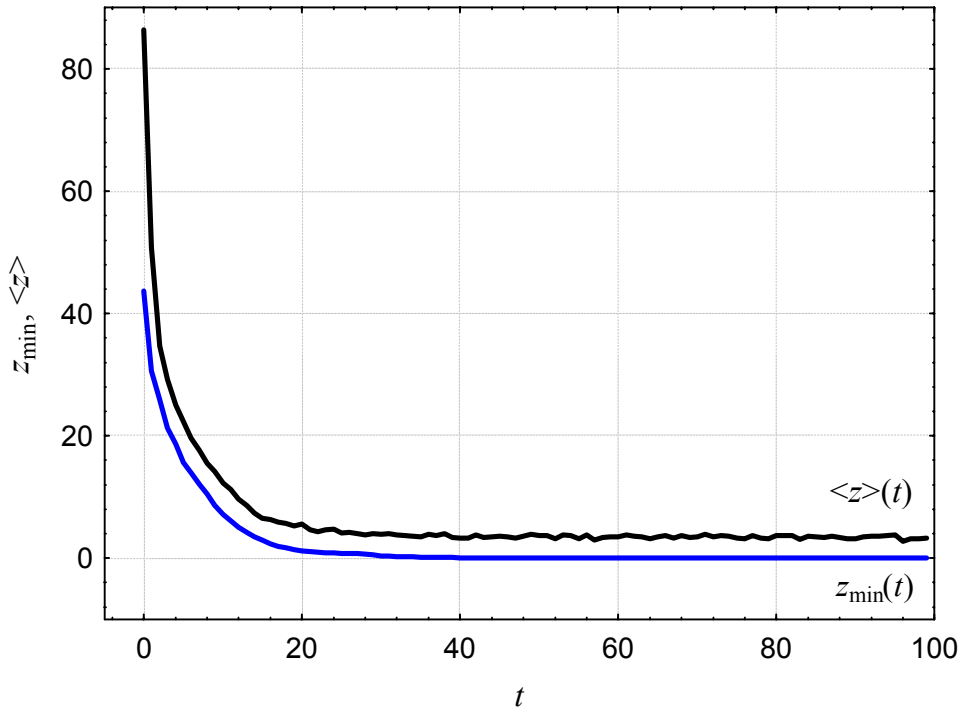

Рис. 9.11. Изменение $z_{min}(t)$ и $<z>(t)$. Популяция из 20 особей, селекция усечением (l = 0,5), одноточечный кроссовер (P_C = 0,7), битовая мутация (P_M = 0,01).

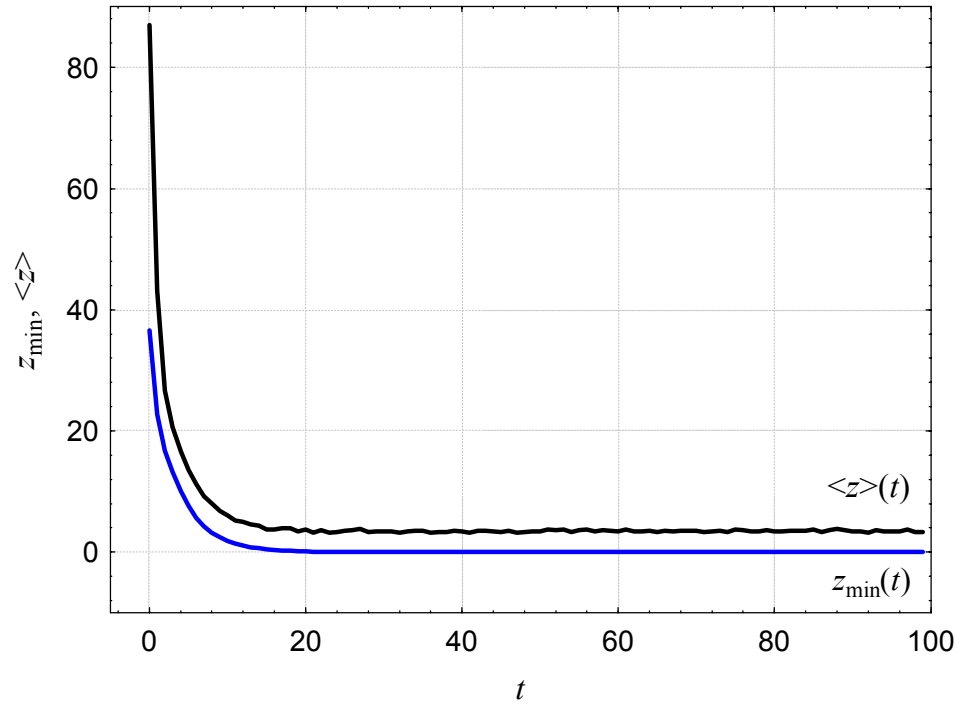

Рис. 9.12. Изменение $z_{min}(t)$ и $<z>(t)$. Популяция из 50 особей, селекция усечением (l = 0,5), одноточечный кроссовер (P_C = 0,7), битовая мутация (P_M = 0,01).

9.7. Общие рекомендации к программной реализации генетического алгоритма

Поскольку генетические алгоритмы представляют собой достаточно универсальный подход к решению экстремальных задач, то выбор языка программирования не играет большой роли.

Саму программу можно писать, используя как объектно-ориентированный подход, так и структурный. Ниже предлагается способ реализации различных компонентов генетического алгоритма при использовании обоих подходов (Табл. 9.2).

Таблица 9.2. Вариант реализации компонентов ГА

Компонент генетического алгоритма	**Структурный подход**	**Объектно-ориентированный подход**
Особь	Одномерный массив для записи значений генов. Размерность массива совпадает с количеством генов у одной особи (количество генов равно числу настраиваемых параметров)	Класс «Особь», содержащий массив генов.
Популяция	Двумерный массив, в котором *i*-я строка содержит гены *i*-й особи.	Отдельный класс «Популяция», содержащий одномерный массив объектов класса, представляющего особь.
Оценивание популяции	Подпрограмма оценки строк массива популяции в соответствии с выбранной целевой функцией.	Метод управляющего класса, оценивающий популяцию в соответствии с выбранной целевой функцией.
Приспособленность популяции	Одномерный массив, в котором *i*-й элемент соответствует приспособленности *i*-й особи.	Одномерный массив со значениями ошибок особей, входящий в управляющий класс.
Особи, выбранные для скрещивания	Двумерный массив, строки которого соответствуют хромосомам особей, выбранным для скрещивания.	Объект класса «Популяция», содержащий объекты класса «Особь», соответствующие выбранным особям

Реализация скрещивания, мутации, формирования нового поколения.	Подпрограммы, обрабатывающие элементы массива, представляющего популяцию особей, а также популяцию особей, выбранных для скрещивания.	Методы управляющего класса, работающие с основной популяцией и популяцией особей для скрещивания.

Приведенный в табл. 9.2 способ реализации генетического алгоритма не является эталонным и, вполне возможно, далек от идеала. Данные в табл. 9.2 могут служить в качестве «опорных» для конкретной реализации генетического алгоритма. Отметим, что бóльшую гибкость и расширяемость программной реализации не только генетического алгоритма, но и любого другого алгоритма и системы вообще, можно достичь, используя компонентно-ориентированный подход и паттерны проектирования [Gamma et al., 1995].

9.8. Задания для лабораторных работ

1. Аппроксимировать набор точек линейной функцией:
$$y(x) = a \cdot x + b.$$
Вариант А) Использовать целочисленное кодирование.
Вариант Б) Использовать вещественное кодирование.

2. Аппроксимировать набор точек экспоненциальной функцией:
$$y(x) = a \cdot \exp(b \cdot x).$$
Вариант А) Использовать целочисленное кодирование.
Вариант Б) Использовать вещественное кодирование.

3. Найти минимум функции:
$$y(x) = x^2 + 4.$$
Вариант А) Использовать целочисленное кодирование.
Вариант Б) Использовать вещественное кодирование.

4. Найти максимум функции:
$$y(x) = 1/x;\ x \in [-4;\ 0).$$
Вариант А) Использовать целочисленное кодирование.
Вариант Б) Использовать вещественное кодирование.

5. Найти точку перегиба функции:
$$f(x) = (x - 1.5)^3 + 3$$
Вариант А) Использовать целочисленное кодирование.
Вариант Б) Использовать вещественное кодирование.

6. Найти точку пересечения функции с осью Ox.
$$f(x) = \ln(x + 1) - 2.25,\ x > -1$$
Вариант А) Использовать целочисленное кодирование.
Вариант Б) Использовать вещественное кодирование.

7. Сгенерировать с помощью генетического алгоритма слово “МИР”.
8. Найти с помощью генетического алгоритма особь, гены которой соответствуют, в формате RGB, фиолетовому цвету (96, 96, 159).

ЛИТЕРАТУРА

[Beyer, Schwefel, Wegener, 2002] Beyer H.-G., Schwefel H.-P., Wegener I. How to analyse Evolutionary Algorithms. Technical Report No.CI-139/02. – University of Dortmund, Germany, 2002.

[De Jong, 1975] De Jong K. An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation. – University of Michigan, Ann Arbor. – University Microfilms No. 76-9381. – 1975.

[Gamma et al., 1995] Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software, Massachusetts: Addison-Wesley, 1995.

[Heitkotter, Beasly, 2001] Heitkotter J., Beasly D. The Hitch-Hiker's Guide to Evolutionary Computation: A List of Frequently Asked Questions (FAQ). ftp://rtfm.mit.edu:/pub/usenet/news.answers/ai-faq/genetic/

[Holland, 1975] Holland J.H. Adaptation in Natural and Artificial Systems. The University of Michigan Press, 1975.

[Koza, 1992] Koza J. Genetic programming: a paradigm for genetically breeding computer population of computer programs to solve problems. MIT Press, Cambridge, MA, 1992.

[Rechenberg, 1973] Rechenberg I. Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Werkstatt Bionik und Evolutionstechnik, Stuttgart: Frommann-Holzboog, 1973.

[Schwefel, 1977] Schwefel H.-P. Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie // Interdisciplinary Systems Research - 1977. – Vol.26.

[Whitley, 2002] Whitley D.L. Genetic Algorithms and Evolutionary Computing. Van Nostrand's Scientific Encyclopedia 2002.

[Бурцев, 2005] Бурцев М.С. Эволюция кооперации в многоагентной системе // Научная сессия МИФИ-2005. VII всероссийская научно-практическая конференция "Нейроинформатика-2005": Сборник научных трудов. В 2-х частях. Ч.1. М.: МИФИ, 2005 - с.217-224

[Емельянов и др., 2003] Емельянов В.В., Курейчик В.В., Курейчик В.М. Теория и практика эволюционного моделирования. – М.: ФИЗМАТЛИТ, 2003. – 432 с.

[Редько, 2003] Редько В.Г. Эволюционная кибернетика. М. – Наука, 2003. – 156с. – (Информатика: неограниченные возможности и возможные ограничения)

[Фогель и др., 1969] Фогель Л., Оуэнс А., Уолш М. Искусственный интеллект и эволюционное моделирование. М.: Мир, 1969. – 230 с.