Министерство образования Российской Федерации

Томский политехнический университет

В. Г. Спицын, Ю. Р. Цой

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ

Методические указания к выполнению лабораторной работы по курсу "ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ИНФОРМАЦИОННЫХ СИСТЕМАХ"

для студентов направления 654700 специальности 071900 "Информационные системы и технологии" АВТФ ТПУ

Томск 2007

СОДЕРЖАНИЕ

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 [1], описывающие динамику изменения молекулярных последовательностей, так и макроэволюционные модели, используемые в экологии, истории и социологии для исследования экосистем и сообществ организмов [1, 2].

Эволюционные принципы используются не только для моделирования, но и для решения прикладных задач оптимизации. Множество алгоритмов и методов, использующие для поиска решения эволюционный подход, объединяют под общим названием **эволюционные вычисления (ЭВ)** или **эволюционные алгоритмы (ЭА)** [3]. Существуют следующие основные виды ЭА:

– генетический алгоритм [4, 5];
– эволюционное программирование [6];
– эволюционные стратегии [7, 8];
– генетическое программирование [9].

В данной главе будет рассмотрен генетический алгоритм (ГА) как один из самых распространенных эволюционных алгоритмов. Краткое описание видов ЭА приведено в [10].

Круг задач, решаемых с помощью ГА, очень широк. Ниже перечислены некоторые задачи, для решения которых используются ГА

[5, 11, 14]:

- 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.

Сама идея применения эволюционных принципов для машинного обучения присутствует также и в известном труде А. Тьюринга, посвященном проблемам создания «мыслящих» машин [15].

1. 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» [4], а также в диссертации его аспиранта Кеннета Де Йонга [12].

Как уже говорилось выше, ГА используют для работы эволюционные принципы наследственности, изменчивости и естественного отбора. Общая схема ГА представлена на рис. 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. 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 *кроссинговера* создается *потомство*, генетическая 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.

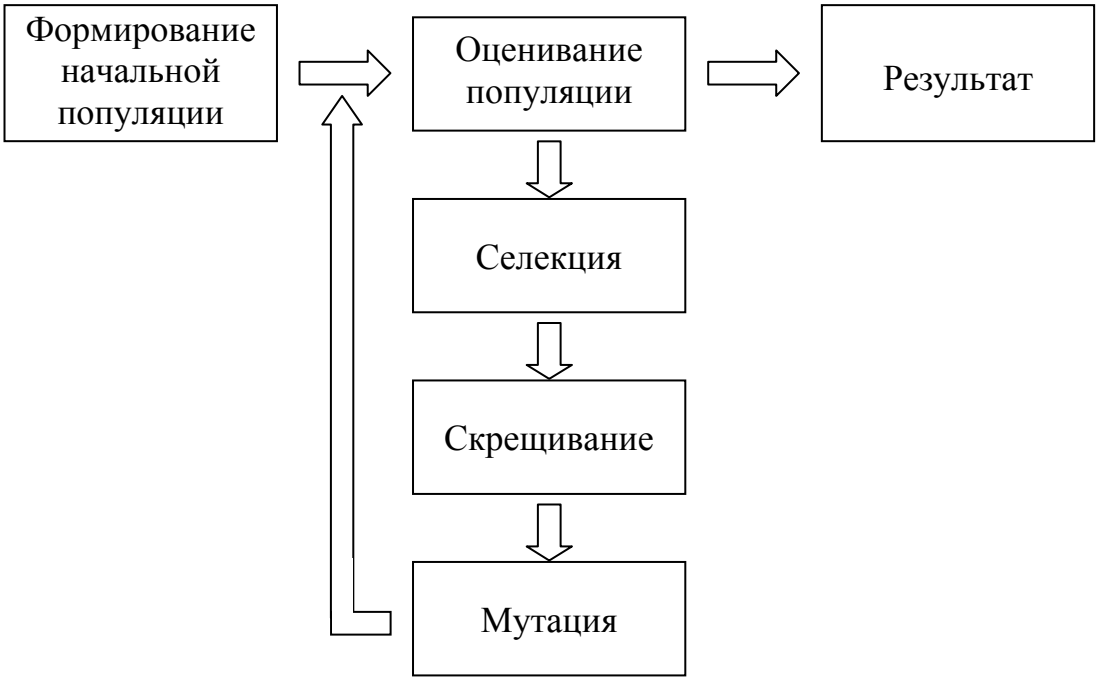

Рис. 1. Общая схема генетического алгоритма

Длительность эволюции может определяться следующими факторами:

- нахождение решения в результате эволюционного поиска;
- ограниченность количества поколений;
- ограниченность количества вычислений функции приспособленности (целевой функции);
- вырождение популяции, когда степень разнородности хромосом в популяции становится меньше допустимого значения.

1.1. Параметры и этапы генетического алгоритма

1.1.1. Кодирование информации и формирование 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 рассматриваемой области 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 классическом генетическом алгоритме хромосома представляет собой битовую строку, в которой закодированы параметры решения поставленной задачи. На рис. 2 показан пример кодирования 4-х 10-разрядных параметров в 40-разрядной хромосоме.

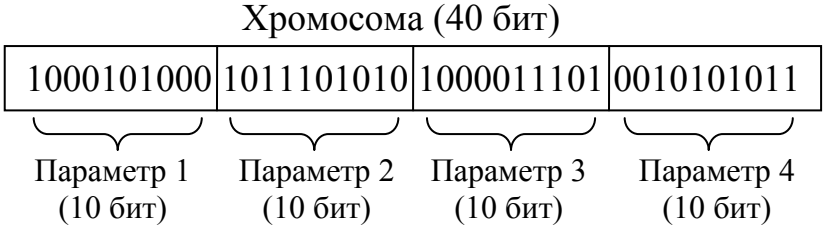

Рис. 2. Пример целочисленного кодирования

Как правило, считают, что каждому параметру соответствует свой *ген*. Таким образом, можно также сказать, что хромосома на рис. 2 состоит из 4-х 10-разрядных генов. Несмотря на то, что каждый параметр закодирован в хромосоме целым числом (в виде двоичной последовательности), 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 $[x_{\min}\ ;\ x_{\max}]$, в пределах которого лежит значение параметра и m – разрядность гена, то этот диапазон разбивают на 2^m равных отрезков, 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.

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. 3.

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

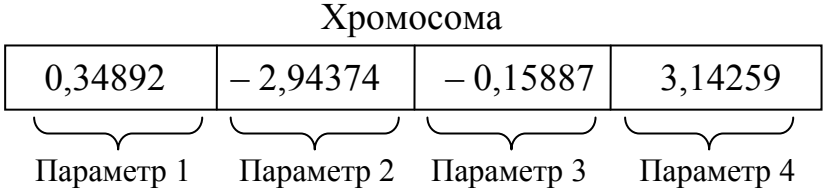

Рис. 3. Пример вещественного кодирования

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

1.1.2. Оценивание популяции

Оценивание популяции необходимо для того, чтобы выявить в

ней более приспособленные и менее приспособленные особи. Для подсчета приспособленности каждой особи используется функция приспособленности (*целевая функция*)

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

где $\mathbf{G}_i = \{ g_{ik} : k = 1,2,...,N \}$ – 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 закодированные в хромосоме целочисленные значения к вещественным числам. Другими словами:

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

где $\mathbf{X}_i = \{ x_{ik} : k = 1,2,...,N \}$ – 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 и $f(\mathbf{G}_i) < f(\mathbf{G}_j)$, 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 считается более приспособленной, чем j-я особь.

1.1.3. 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 особи 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 *l*, 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 $t = 2, 3, 4, 5$.

1.1.4. Скрещивание и формирование нового поколения

Отобранные в результате селекции особи (называемые *родительскими*) скрещиваются и дают потомство. Хромосомы потомков формируются в процессе обмена генетической информацией (с применением оператора *кроссинговера*) между родительскими особями. Созданные таким образом потомки составляют популяцию следующего поколения. Ниже будут описаны основные операторы кроссинговера 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 (P_C > RANDOM) {
        СКРЕЩИВАНИЕ (РОДИТЕЛЬ[i], РОДИТЕЛЬ[j],
                     ПОТОМОК[k], ПОТОМОК[k+1]);
        k = k+2;
    } ИНАЧЕ {
        ПОТОМОК[k] = РОДИТЕЛЬ[i];
```

```
            ПОТОМОК[k+1] = РОДИТЕЛЬ[j];
        }
    }
```

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

1-точечный кроссинговер 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 кроссинговера представлен на рис. 4а.

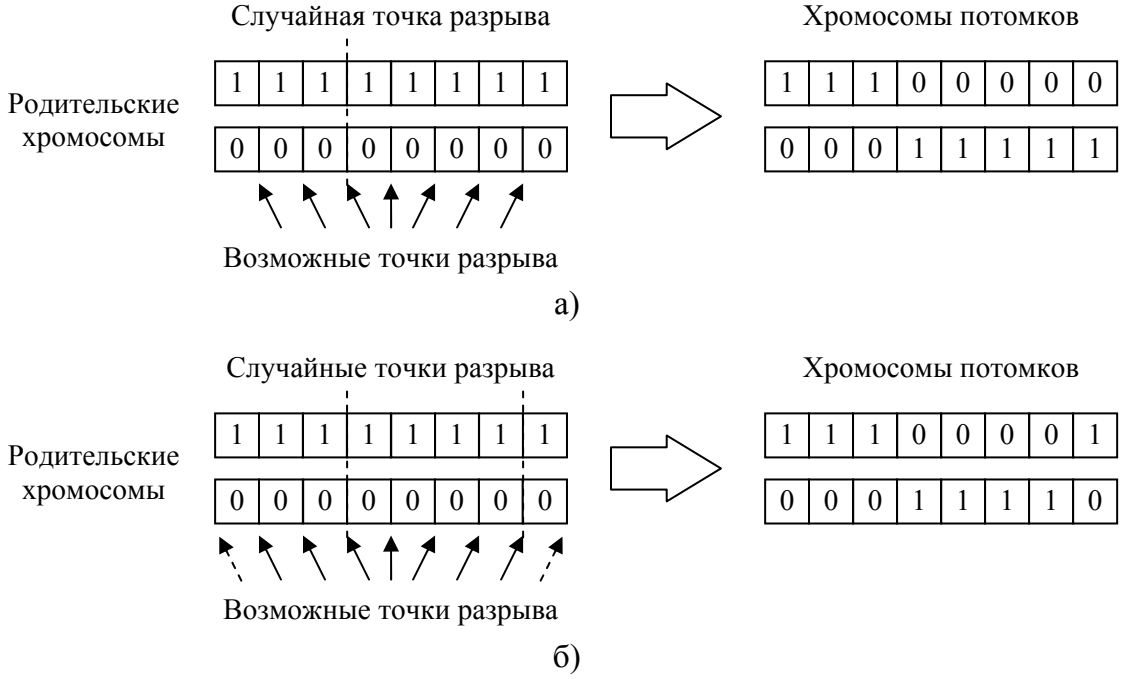

Рис. 4. Примеры работы: а) 1-точечного оператора кроссинговера; б) 2-точечного оператора кроссинговера.

Для оператора *2-точечного кроссинговера* выбираются 2 случайные точки разрыва, после чего для создания потомков родительские хромосомы обмениваются участками, лежащими между точками разрыва (рис. 4б). Отметим, что для 2-точечного оператора кроссинговера, начало и конец хромосомы считаются «склеенными» в результате чего одна из точек разрыва может попасть в начало/конец хромосом и в таком случае результат работы 2-точечного кроссинговера будет совпадать с результатом работы 1-точечного кроссинговера [12]. На рис. 4б точка разрыва в месте склеивания хромосом показана пунктирными

стрелками.

При использовании *однородного оператора кроссинговера* разряды родительских хромосом наследуются 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 $(1 - p_0)$. Каждый разряд родительских хромосом «разыгрывается» в соответствии со значением p_0 между хромосомами потомков. 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 кроссинговера.

2-точечный кроссинговер для вещественного кодирования в целом аналогичен 2-точечному кроссинговеру для целочисленного кодирования. Различие заключается в том, что точка разрыва не может быть выбрана «внутри» гена, а должна попасть между генами (рис. 5).

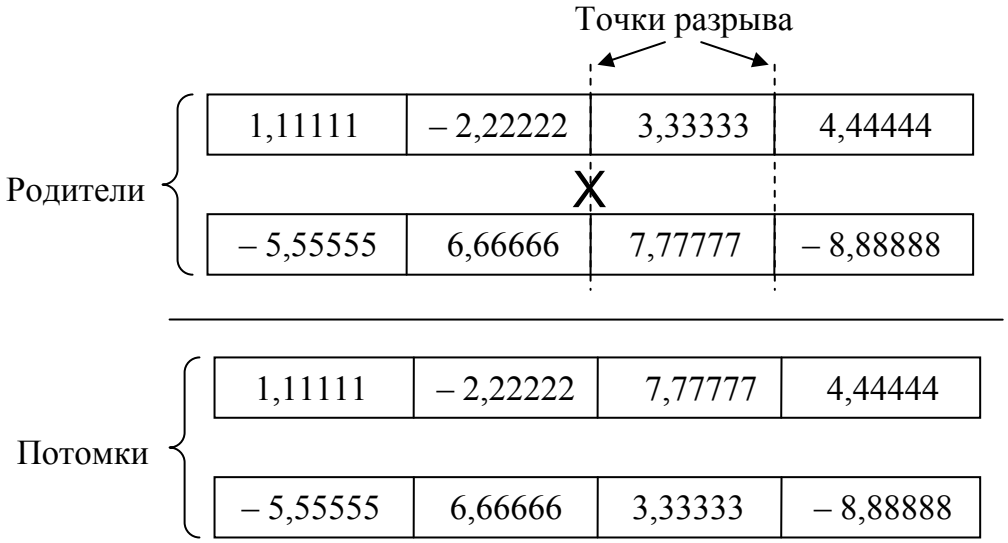

Рис. 5. Пример работы 2-точечного кроссинговера для вещественного кодирования

При использовании арифметического и *BLX-*α операторов кроссинговера 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 кроссинговера:

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

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

где $0 \le \lambda \le 1$.

Если используется *BLX*-α кроссинговер, 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 кроссинговера, показано на рис. 6.

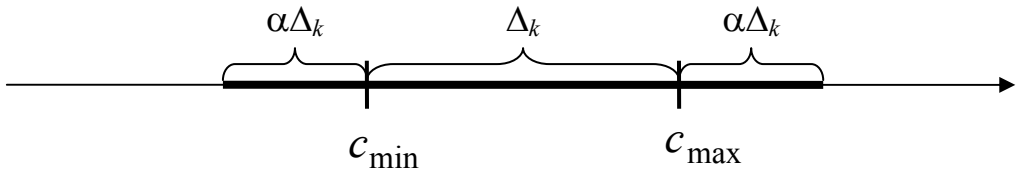

Рис. 6. Интервал для *BLX*-α кроссинговера

Разрушающая способность кроссинговера. Операторы кроссинговера характеризуются *способностью к разрушению* (*dispurtion*) родительских хромосом. Кроссинговер 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 кроссинговера к разрушению зависит от того, насколько сильно он «перемешивает» (рекомбинирует) содержимое родительских хромосом. Так, 1-точечный кроссинговер считается слаборазрушающим, а однородный кроссинговер в большинстве случаев является сильно разрушающим оператором. Соответственно, 2-точечный кроссинговер по разрушающей способности занимает промежуточную позицию по отношению к 1-точечному и однородному операторам кроссинговера.

В случае кроссинговера 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 кроссинговера зависит от содержимого родительских хромосом. Разрушающая способность арифметического кроссинговера зависит от значения параметра λ, например, при $\lambda \to 1$ и $\lambda \to 0$, способность к разрушению будет низкой. Для *BLX-*α кроссинговера 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*) кроссинговером новых особей. Тем самым подчеркивается, что, разрушая хромосомы родительских особей, кроссинговер 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. Пусть доля обновляемых особей равна T, $0 < T < 1$, тогда 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*) [12]. Использование элитных особей позволяет увеличить скорость сходимости генетического алгоритма.

1.1.5. Мутация

Оператор *мутации* используется для внесения случайных изменений в хромосомы особей. Это позволяет «выбираться» из локальных экстремумов и тем самым эффективнее исследовать пространство поиска. Аналогично оператору кроссинговера, работа оператора мутации зависит от вероятности применения мутации P_M.

Рассмотрим базовые варианты оператора мутации в зависимости от способа представления генетической информации.

Целочисленное кодирование. Одним из основных операторов мутации для целочисленного кодирования является битовая мутация. 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 [– ξ; + ξ], 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. Также возможна адаптивная подстройка величины диапазона 2ξ изменения значения гена в результате мутации.

2. НАСТРОЙКА ПАРАМЕТРОВ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

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

– длительность эволюции (количество поколений);

– размер популяции;

– интенсивность (давление) селекции;
– тип оператора кроссинговера;
– вероятность кроссинговера P_C;
– тип 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 влияние параметров на эволюционный процесс имеют сложный характер. На рис. 7 схематично изображено влияние изменения некоторых параметров ГА на характеристики эволюционного поиска.

Неправильная настройка параметров может стать причиной различных проблем в работе ГА. Краткий список часто встречающихся проблем и возможные пути их исправления приведены в табл. 1.

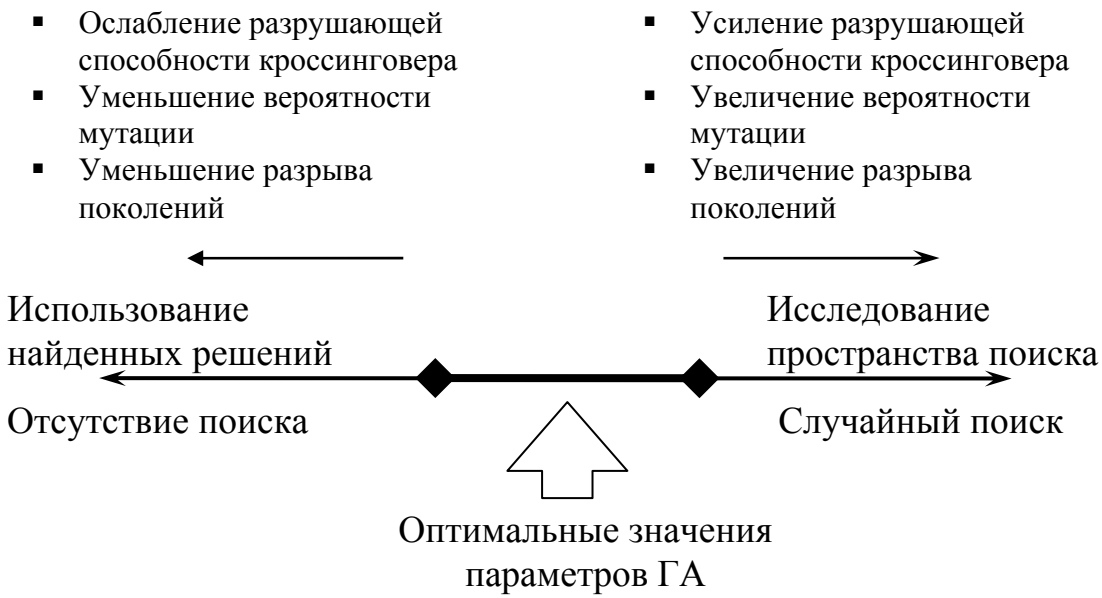

Рис. 7. Влияние параметров ГА на характеристики эволюционного поиска

Табл. 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 популяции)	1. Изменение стратегии выбора родительских пар для скрещивания. 2. Отслеживание появления в популяции идентичных особей и их удаление. 3. Использование сильно разрушающего оператора кроссинговера. 4. Увеличение вероятности мутации.

3. Низкая «стабильность» эволюции популяции (значительные колебания значения средней приспособленности от поколения к поколению)	1. Применение “элитизма” (уменьшение разрыва поколений). 2. Уменьшение вероятности мутации. 3. Использование кроссинговера 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.

3. КАНОНИЧЕСКИЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ

Канонический генетический алгоритм разработан Джоном Холландом и описан в его книге «Адаптация в естественных и искусственных системах», 1975 г. [4]. Представляет одну из базовых моделей эволюционного поиска, подробно исследованную в 70-80-х годах 20 века. Канонический ГА имеет следующие характеристики:

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

4. ПРИМЕР РАБОТЫ И АНАЛИЗА ГЕНЕТИЧЕСКОГО АЛГОРИТМА

При использовании генетического алгоритма для решения задачи оптимизации необходимо:

1. Определить количество и тип оптимизируемых 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. Определить параметры 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. Для анализа результатов работы ГА необходимо произвести несколько запусков алгоритма, для повышения достоверности выводов о качестве получаемых результатов, т.к. результат работы ГА носит вероятностный характер. Описанная общая схема решения задачи с использованием ГА показана на рис. 8.

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

$$z = \sum_{i=1}^{n} x_i^2, n = 10, x_i \in [-5{,}12;\ 5{,}11], \quad (1)$$

$$z \to \min.$$

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. 8):

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 хромосоме 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:

$$f_i = z_i,$$

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

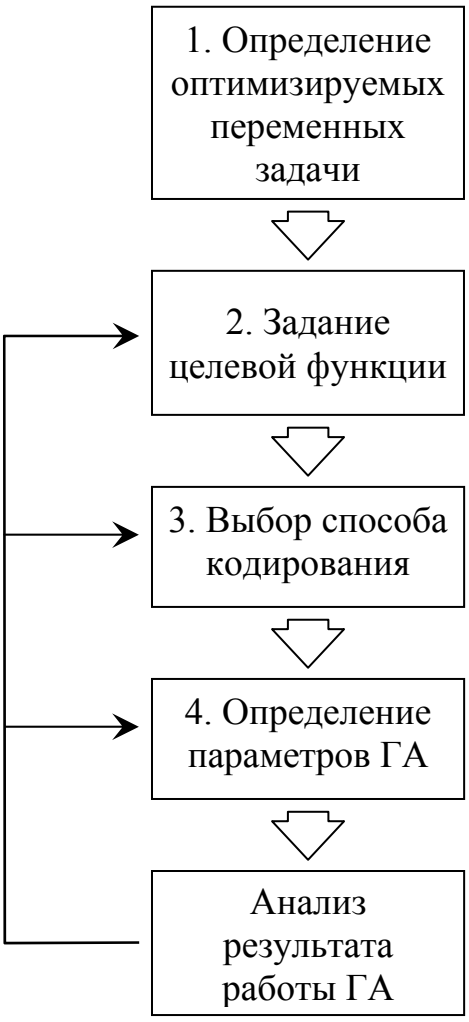

Рис. 8. Общая схема решения задачи с использованием ГА

3. Выбор способа кодирования. В качестве способа представления генетической информации рассмотрим целочисленное кодирование с точностью кодирования параметров 0,01. Тогда в имеющемся по условию задачи диапазоне изменения значений параметров [–5,12; 5,11] можно закодировать (5,12 – (-5,11))/0,01 + 1 = 1024 различных значений переменной. Единица прибавляется, так как значение переменной равное 0 также учитывается.

Для того чтобы представить 1024 различных значений переменной, достаточно использовать $\log_2 1024 = 10$ бит на каждую переменную. Таким образом, будет использоваться целочисленное кодирование с 10-разрядными генами.

4. Определение параметров ГА. Для решения задачи рассмотрим популяцию из 20 особей. При отборе особей для скрещивания будем

использовать турнирную селекцию с бинарным турниром. В качестве генетических операторов будем использовать 1-точечный кроссинговер и битовую мутацию. Вероятности применения операторов скрещивания и мутации установим равными 0,7 и 0,05, 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. Показаны зависимости изменения среднего $<z>$ и наименьшего $z_{\min}$ в популяции значения функции z от номера поколения t. Данные усреднены по 100 независимым запускам.

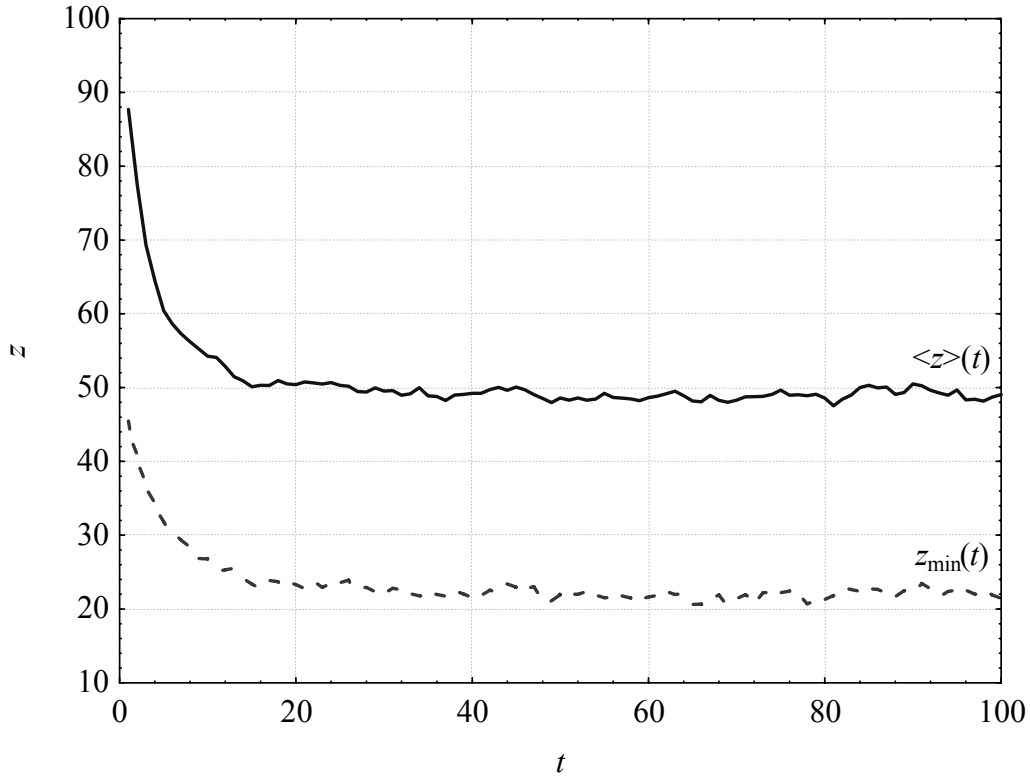

Рис. 9. Изменение $z_{\min}(t)$ и $<z>(t)$. Популяция из 20 особей, бинарный турнирный отбор, одноточечный кроссинговер ($P_C = 0{,}7$), битовая мутация ($P_M = 0{,}05$)

По данным рис. 9 видно, что после 20-го 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. 10.

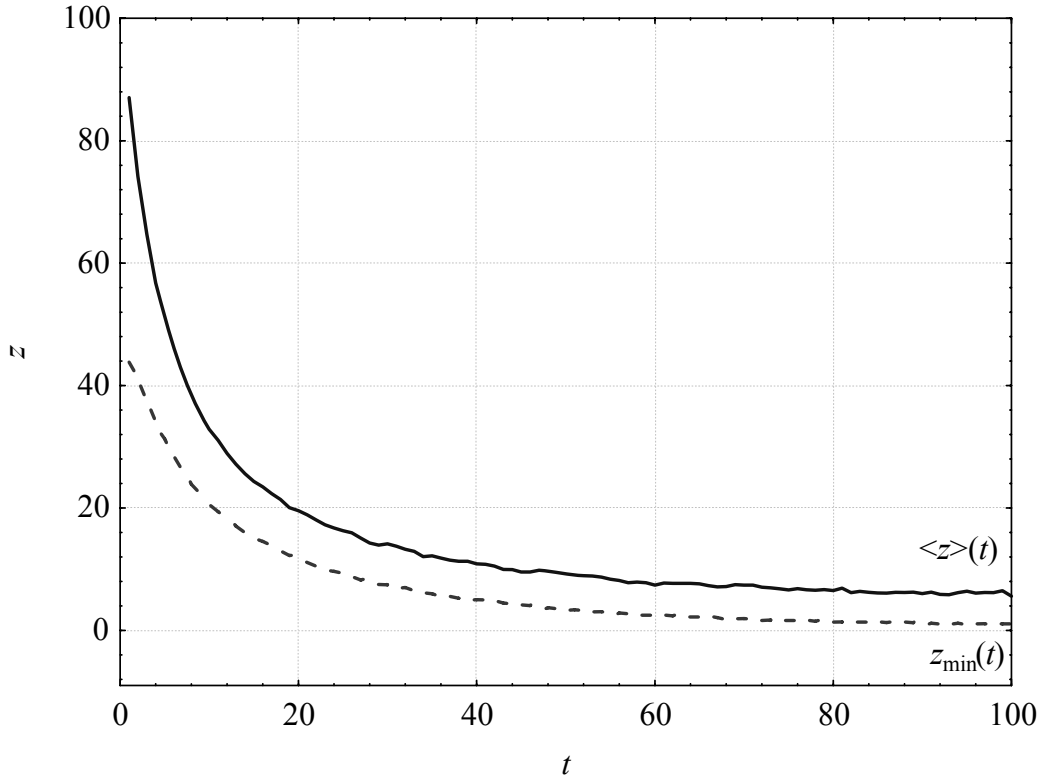

Рис. 10. Изменение $z_{\min}(t)$ и $<z>(t)$. Популяция из 20 особей, бинарный турнирный отбор, одноточечный кроссинговер ($P_C = 0{,}7$), битовая мутация ($P_M = 0{,}01$)

Из сравнения графиков на рис 9 и 10 следует, что уменьшение вероятности мутации улучшило результат работы ГА. Также отметим, что теперь эволюционный процесс стабилизировался значительно позднее, примерно после 60-го поколения. Усредненное по всем запускам минимальное значение функции z, достигнутое за первые 100 поколений, равно ~1,016. Чтобы улучшить результат, увеличим давление селекции путем увеличения размера турнира до 4. Результат представлен на рис. 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-го поколения, а усредненное по всем запускам минимальное полученное значение функции z равно ~0,013. 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 0,01, для ГА с параметрами, соответствующими графикам на рис. 11, решение было найдено в 69 запусках из 100. При этом в среднем было использовано 1698,68 вычислений целевой

функции.

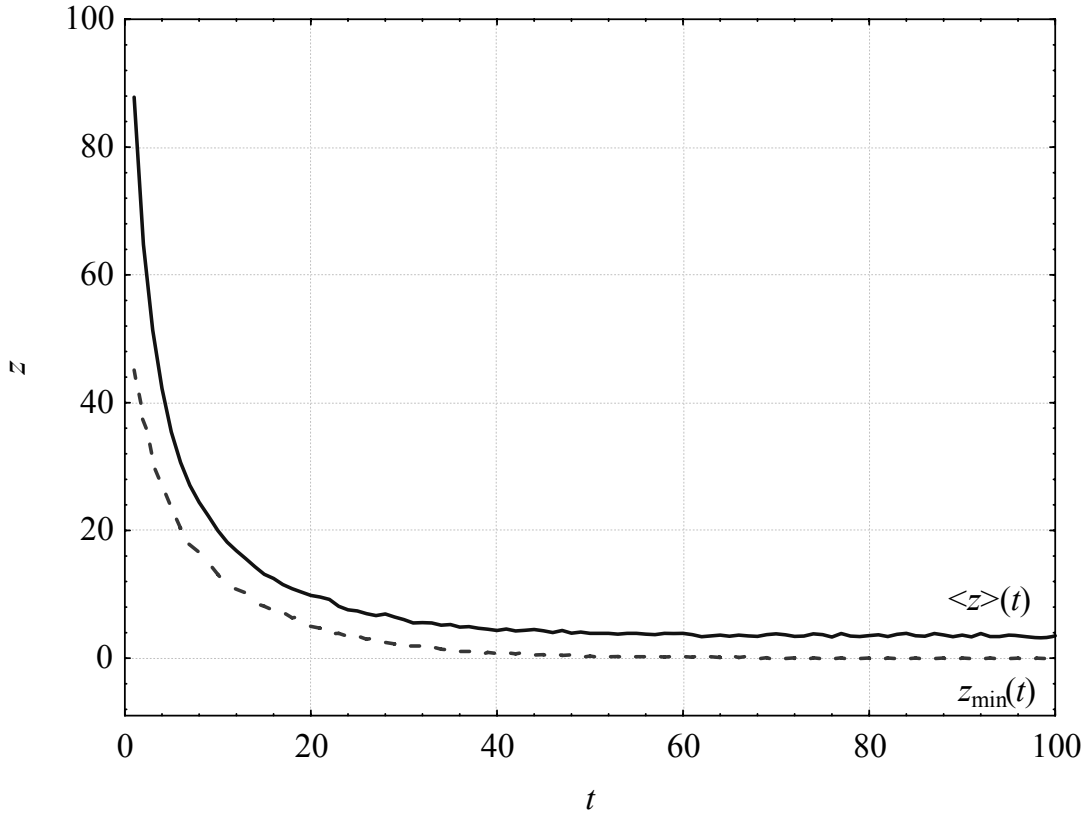

Рис. 11. Изменение $z_{\min}(t)$ и $\langle z \rangle(t)$. Популяция из 20 особей, турнирный отбор ($t = 4$), одноточечный кроссинговер ($P_C = 0{,}7$), битовая мутация ($P_M = 0{,}01$)

Чтобы повысить стабильность результатов, увеличим размер популяции до 50 особей. Полученные кривые $z_{\min}(t)$ и $\langle z \rangle(t)$ изображены на рис. 12. Во всех 100 запусках найден минимум функции z с точностью не меньше 0,01. Среднее количество вычислений целевой функции, использованное для нахождения решения, равно 3145,34.

5. ОБЩИЕ РЕКОМЕНДАЦИИ К ПРОГРАММНОЙ РЕАЛИЗАЦИИ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Программную реализацию ГА можно создать, используя как объектно-ориентированный, так и структурный подход. Ниже предлагается способ реализации различных компонентов генетического алгоритма с использованием обоих подходов (табл. 2).

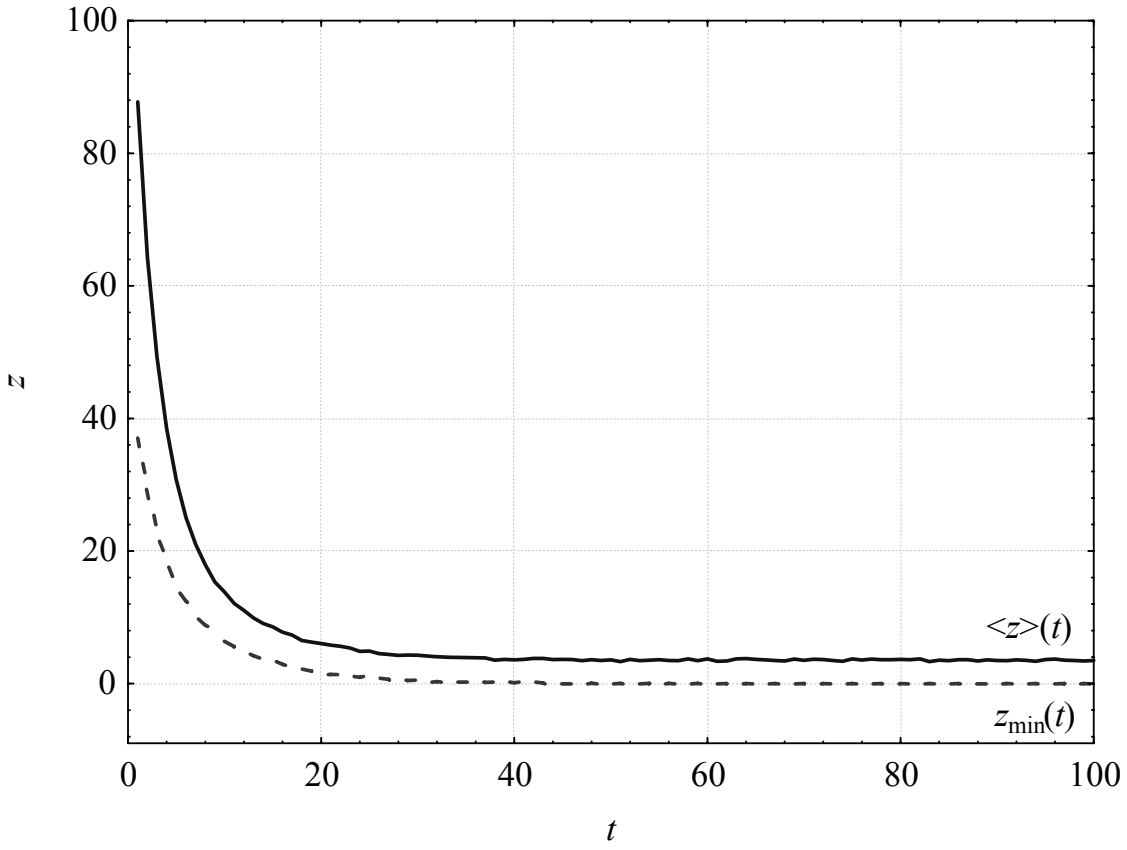

Рис. 12. Изменение $z_{min}(t)$ и $<z>(t)$. Популяция из 50 особей, турнирный отбор ($t = 4$), одноточечный кроссинговер ($P_C = 0{,}7$), битовая мутация ($P_M = 0{,}01$)

Приведенный в табл. 2 способ реализации генетического алгоритма не является эталонным и, вполне возможно, далек от идеала. Данные в табл. 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 [13, 16].

Табл. 2. Варианты 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

6. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ

Целью лабораторной работы является создание студентом программы, реализующей генетический алгоритм для решения задачи оптимизации.

Отчет по лабораторной работе должен содержать:

1. Цель работы.
2. Постановку задачи.
3. Метод решения задачи.
4. Структурную схему алгоритма.
5. Листинг программы.
6. Результаты работы генетического алгоритмы.
7. Выводы.

Для создания экспертной системы рекомендуется использование языков программирования: Турбо-Пролога, C ++, JAVA, Delphi, C#.

Ниже излагаются варианты заданий для выполнения лабораторных работ. Выбор варианта производится в соответствии с желанием студента, на основании его знаний о предметной области.

7. 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:
$$y(x) = 1/x;\ \text{x} \in [-4;\ 0).$$
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

1. Редько В.Г. Эволюционная кибернетика. М. – Наука, 2003. – 156 с.
2. 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.
3. Beyer H.-G., Schwefel H.-P., Wegener I. How to analyse Evolutionary Algorithms. Technical Report No.CI-139/02. – University of Dortmund, Germany, 2002.
4. Holland J.H. Adaptation in Natural and Artificial Systems. The University of Michigan Press, 1975.
5. Емельянов В.В., Курейчик В.В., Курейчик В.М. Теория и практика эволюционного моделирования. – М.: ФИЗМАТЛИТ, 2003. – 432 с.
6. Фогель Л., Оуэнс А., Уолш М. Искусственный интеллект и эволюционное моделирование. М.: Мир, 1969. – 230 с.
7. Rechenberg I. Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Werkstatt Bionik und Evolutionstechnik, Stuttgart: Frommann-Holzboog, 1973.
8. Schwefel H.-P. Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie // Interdisciplinary Systems Research: – 1977. – Vol. 26.
9. Koza J. Genetic programming: a paradigm for genetically breeding computer population of computer programs to solve problems. MIT Press, Cambridge, MA, 1992.
10. Whitley D.L. Genetic Algorithms and Evolutionary Computing. Van Nostrand's Scientific Encyclopedia 2002.

11. 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/genetшc/.
12. 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.
13. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software, Massachusetts: Addison-Wesley, 1995.
14. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы / Под ред. В.М. Курейчика. – 2-е изд., испр. и доп. – М.: Физматлит, 2006. – 320 с.
15. Turing A. M. Computing machinery and intelligence // Mind, 1950, vol. 236, no. 59.
16. Цой Ю.Р. ECWorkshop – инструментальная библиотека классов для эволюционных вычислений // Труды международных научно-технических конференций «Интеллектуальные системы (IEEE AIS'07)» и «Интеллектуальные САПР (CAD-2007)». – М.: Физматлит, 2007. – С.94-101.