ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Т.В. Панченко

Генетические алгоритмы

*Допущено Учебно-методическим объединением
по образованию в области математических методов
в экономике в качестве учебно-методического пособия для студентов
высших учебных заведений, обучающихся по специальности
«Математические методы в экономике»
и другим математическим специальностям.*

Под редакцией Ю.Ю. Тарасевича

Издательский дом «Астраханский университет»
2007

ББК 32.97 : 22.12
П16

Панченко, Т. В. Генетические алгоритмы [Текст] : учебно-методическое пособие / под ред. Ю. Ю. Тарасевича. — Астрахань : Издательский дом «Астраханский университет», 2007. — 87 [3] с.

Рецензенты:

доктор физико-математических наук, профессор В. А. Еремеев (Ростовский государственный университет);
кандидат технических наук, доцент Я. М. Русанова (Ростовский государственный университет);
кандидат технических наук, доцент В. Я. Трофимец (Ярославский военный финансово-экономический институт)

В книге рассмотрены генетические алгоритмы, широко применяемые в последнее время для решения задач оптимизации. Описываются стандартные функции универсального пакета MATLAB 7.0.1, предназначенные для решения задач оптимизации с помощью генетических алгоритмов. Приводится лабораторная работа, посвященная изучению методов решения оптимизационных задач с помощью MATLAB 7.0.1.

Предназначено для студентов V курса, обучающихся по специальности «Прикладная математика и информатика».

ISBN 5-88200-913-8

Содержание

1. Введение **5**

2. Генетические алгоритмы **6**
2.1. Простой пример генетического алгоритма 6
2.2. Основные понятия 12
2.3. Операторы выбора родителей 15
2.4. Рекомбинация (воспроизведение) 18
2.4.1. Дискретная рекомбинация 18
2.4.2. Кроссинговер (бинарная рекомбинация) 20
2.5. Мутация 23
2.6. Операторы отбора особей в новую популяцию 26
2.7. Разнообразие генетических алгоритмов 28
2.7.1. Канонический ГА 28
2.7.2. Генитор 29
2.7.3. Метод прерывистого равновесия 29
2.7.4. Гибридный алгоритм 29
2.7.5. CHC 30
2.7.6. ГА с нефиксированным размером популяции 30
2.8. Параллельное выполнение ГА 31
2.8.1. Параллельный ГА 31
2.8.2. Миграция 32
2.8.3. Глобальная модель «Рабочий и Хозяин» 34
2.8.4. Модель диффузии, или островная модель ГА 35

3. Параметры ГА **37**

4. Модернизация ГА **37**
4.1. Самоадаптирующиеся алгоритмы 38
4.1.1. Неоднородная мутация 38
4.1.2. Инцест 38
4.1.3. Критерий расстояния 39
4.1.4. Параметры вычислений 40

5. Символьная модель ГА **41**
5.1. Постановка задачи 41
5.2. Символьная модель 41
5.3. Геометрическая интерпретация символьной модели 43
5.4. Шима 44
5.5. Строительные блоки 48

5.6. Теорема шим 48

6. Преимущества и недостатки ГА 50

7. Заключение 54

A. Приложение 55
A.1. Теория Дарвина 55
A.2. Некоторые понятия из теории оптимизации 56
A.3. Кодирование Грея 58
A.3.1. Строение кода Грея 58
A.3.2. Алгоритмы преобразования кода Грея 58
A.4. NP-полные (универсальные) задачи 59
A.5. Тестовые функции 60

B. Генетические алгоритмы в MATLAB 7.0.1 66
B.1. Общие сведения 66
B.2. Функция ga 66
B.3. Функция gaoptimset 69
B.4. Векторизация целевой функции 73
B.5. Лабораторная работа 75
B.5.1. Лабораторная работа № 1 75
B.5.2. Лабораторная работа № 2 79
B.5.3. Лабораторная работа № 3 81
B.6. Ответы 81

Библиографический список 86

1. Введение

Данная работа посвящена одному из оптимизационных методов. Как известно, оптимизационные задачи заключаются в нахождении минимума (максимума) заданной функции. Такую функцию называют целевой. Как правило, целевая функция — сложная функция, зависящая от некоторых входных параметров. В оптимизационной задаче требуется найти значения входных параметров, при которых целевая функция достигает минимального (максимального) значения. Существует целый класс оптимизационных методов. Условно все оптимизационные методы можно разделить на методы, использующие понятие производной (градиентные методы) и стохастические методы (например, методы группы Монте-Карло). С их помощью можно найти экстремальное значение целевой функции, но не всегда можно быть уверенным, что получено значение глобального экстремума. Нахождение локального экстремума вместо глобального называется преждевременной сходимостью. Помимо проблемы преждевременной сходимости существует другая проблема — время процесса вычислений. Зачастую более точные оптимизационные методы работают очень долго.

Для решения поставленных проблем и проводится поиск новых оптимизационных алгоритмов. Предложенные сравнительно недавно — в 1975 году — Джоном Холландом генетические алгоритмы (ГА) основаны на принципах естественного отбора Ч. Дарвина. ГА относятся к стохастическим методам. Эти алгоритмы успешно применяются в различных областях деятельности (экономика, физика, технические науки и т.п.). Созданы различные модификации ГА и разработан ряд тестовых функций. Рассмотреть как работают ГА, и какие проблемы остаются неразрешенными — цель данной работы.

Генетические алгоритмы относят к области мягких вычислений. Термин «мягкие вычисления» введен Лофти Заде в 1994 году [**?**]. Это понятие объединяет такие области, как нечеткая логика, нейронные сети, вероятностные рассуждения, сети доверия[1] и эволюционные алгоритмы, которые дополняют друг друга и используются в различных комбинациях или самостоятельно для создания гибридных интеллектуальных систем. Первая схема генетического алгоритма была предложена в 1975 году в Мичиганском университете Джоном Холландом (John Holland) [8], а предпосыл-

[1] web of trust — концепция построения распределенной сетевой инфраструктуры, в которой нет необходимости иметь универсальные для всех корневые (т.е. те ключи, которым доверяют по умолчанию) ключи кодирования и декодирования информации, и каждая из сторон может доверять своему набору корневых ключей (например, своему собственному ключу и ключам, ею подписанным).

ками этому послужили работы Ч. Дарвина [2] (теория эволюции) и исследования Л.Дж.Фогеля, А.Дж. Оуэнса, М.Дж.Волша [?] по эволюции простых автоматов, предсказывающих символы в цифровых последовательностях (1966)[2]. Новый алгоритм получил название «репродуктивный план Холланда» и в дальнейшем активно использовался в качестве базового алгоритма в эволюционных вычислениях. Идеи Холланда развили его ученики Кеннет Де Йонг (Kenneth De Jong) из университета Джорджа Мейсона (Вирджиния) [22] и Дэвид Голдберг (David E. Goldberg) из лаборатории ГА Иллинойса [6]. Благодаря им, был создан классический ГА, описаны все операторы и исследовано поведение группы тестовых функций (именно алгоритм Голдберга и получил название «генетический алгоритм»).

Генетические алгоритмы — это адаптивные методы поиска, которые в последнее время используются для решения задач оптимизации. В них используются как аналог механизма генетического наследования, так и аналог естественного отбора. При этом сохраняется биологическая терминология в упрощенном виде и основные понятия линейной алгебры.

2. Генетические алгоритмы

2.1. Простой пример генетического алгоритма

Рассмотрим принципы работы генетических алгоритмов на максимально простом примере.

Пусть требуется найти глобальный минимум функции

$$y(x) = 5 - 24x + 17x^2 - \frac{11}{3}x^3 + \frac{1}{4}x^4$$

на отрезке $[0; 7]$ (рис. 1). На этом отрезке функция принимает минимальное значение в точке $x = 1$. Очевидно, что в точке $x = 6$ функция попадает в локальный минимум. Если для нахождения глобального минимума использовать градиентные методы, то в зависимости от начального приближения можно попасть в данный локальный минимум.

Рассмотрим на примере данной задачи принцип работы генетических алгоритмов. Для простоты положим, что x принимает лишь целые значения, т.е. $x \in \{0, 1, 2, 3, 4, 5, 6, 7\}$. Это предположение существенно упростит

[2]Простые конечные автоматы преобразуют некоторую входную числовую последовательность согласно своему строению. Они широко используются в области распознавания образов и языка. Для построения таких автоматов используют эволюционные методы.

изложение, сохранив все основные особенности работы генетического алгоритма.

Выберем случайным образом несколько чисел на отрезке $[0;7]$: $\{2,3,5,4\}$. Будем рассматривать эти числа в качестве пробных решений нашей задачи.

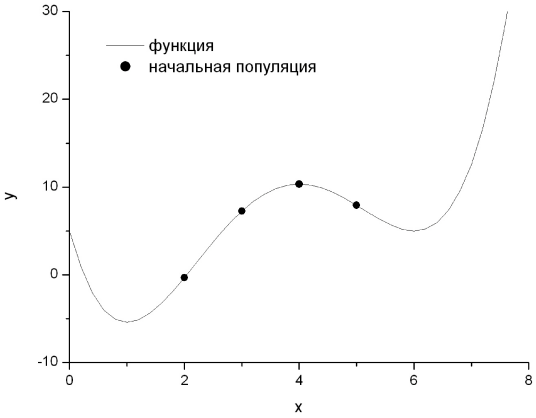

Рис. 1. График целевой функции с выбранными значениями пробных решений

Основной идеей генетических алгоритмов является организация «борьбы за существование» и «естественного отбора» среди этих пробных решений. Запишем пробные решения в двоичной форме: $\{010, 011, 101, 100\}$. Поскольку генетические алгоритмы используют биологические аналогии, то и применяющаяся терминология напоминает биологическую. Так, одно пробное решение, записанное в двоичной форме, мы будем называть *особью* или *хромосомой*, а набор всех пробных решений – *популяцией*. Как известно, принцип естественного отбора заключается в том, что в конкурентной борьбе выживает наиболее приспособленный. В нашем случае приспособленность особи определяется целевой функцией: чем меньше значение целевой функции, тем более приспособленной является особь, т.е. пробное решение, использовавшееся в качестве аргумента целевой функции (см. табл. 1).

Теперь приступим к процессу *размножения*: попробуем на основе исходной популяции создать новую, так чтобы пробные решения в новой популяции были бы ближе к искомому глобальному минимуму целевой функ-

ции. Для этого сформируем из исходной популяции *брачные пары* для скрещивания. Поставим в соответствие каждой особи исходной популяции случайное целое число из диапазона от 1 до 4. Будем рассматривать эти числа как номера членов популяции. При таком выборе какие-то из членов популяции не будут участвовать в процессе размножения, так как образуют пару сами с собой. Какие-то члены популяции примут участие в процессе размножения неоднократно с различными особями популяции. Процесс размножения (*рекомбинация*) заключается в обмене участками хромосом между родителями. Например, пусть скрещиваются две хромосомы 111111 и 000000. Определяем случайным образом точку разрыва хромосомы, пусть, это будет 3: 111|111 000|000. Теперь хромосомы обмениваются частями, стоящими после точки разрыва, и образуют двух новых потомков: 111000 и 000111.

Таблица 1

Исходная популяция

№	Особь		
	Целое число	Двоичное число	Приспособленность
1	2	010	−0,33
2	3	011	7,25
3	5	101	7,92
4	4	100	10,33

Для нашей популяции процесс создания первого поколения потомков показан в таблице 2.

Таблица 2

Одноточечный кроссинговер

№	Особь популяции	Выбранный номер	Вторая особь-родитель	Точка кроссинговера	Особи-потомки
1	010	1	010	1	000
2	011	4	100		110
3	101	3	101	2	100
4	100	1	010		011

Следующим шагом в работе генетического алгоритма являются мутации, т.е. случайные изменения полученных в результате скрещивания хромосом. Пусть вероятность мутации равна 0,3. Для каждого потомка возьмем случайное число на отрезке $[0; 1]$, и если это число меньше 0,3, то

инвертируем случайно выбранный ген (заменим 0 на 1 или наоборот) (см. табл. 3).

Таблица 3

Мутация потомков

№	Особи-потомки	Случайное число	Выбранный ген для мутации	Потомок после мутации	Приспособленность потомка до мутации	Приспособленность потомка после мутации
1	000	0,1	3	001	5	
1	000	0,1	3	001	5	−5,42
2	110	0,6	–	110	5	5
3	100	0,5	–	100	10,33	10,33
4	011	0,2	1	111	7,25	12,58

Как видно на примере, мутации способны улучшить (первый потомок) или ухудшить (четвертый потомок) приспособленность особи-потомка. В результате скрещивания хромосомы обмениваются «хвостами», т.е. младшими разрядами в двоичном представлении числа. В результате мутаций изменению может подвергнуться любой разряд, в том числе, старший. Таким образом, если скрещивание приводит к относительно небольшим изменениям пробных решений, то мутации могут привести к существенным изменениям значений пробных решений (см. рис. 2).

Теперь из четырех особей-родителей и четырех полученных особей потомков необходимо сформировать новую популяцию. В новую популяцию отберем четыре наиболее приспособленных особей из числа «старых» особей и особей-потомков (см. табл. 4).

В результате получим новое поколение, которое представлено на рис. 3.

Получившуюся популяцию можно будет вновь подвергнуть кроссинговеру, мутации и отбору особей в новое поколение. Таким образом, через несколько поколений мы получим популяцию из похожих и наиболее приспособленных особей. Значение приспособленности наиболее «хорошей» особи (или средняя приспособленность по популяции) и будет являться решением нашей задачи. Следуя этому, в данном случае, взяв наиболее

приспособленную особь 001 во втором поколении, можно сказать, что минимумом целевой функции является значение $-5{,}42$, соответствующее аргументу $x = 1$. Тем самым попадания в локальный минимум удалось избежать! На данном примере разобран вариант простого генетического алгоритма. При дальнейшем использования ГА к разным задачам возможно моделирование основных операторов алгоритма.

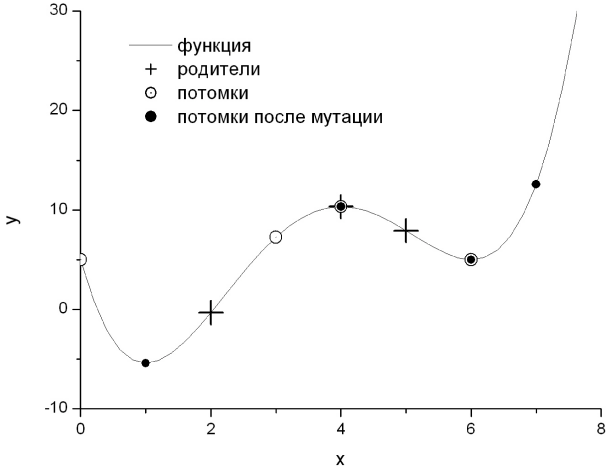

Рис. 2. Изменение популяции в процессе естественного отбора

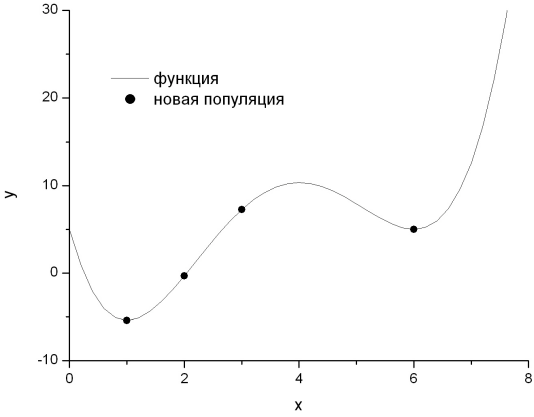

Рис. 3. Минимизируемая функция и особи новой популяции

Таблица 4

Формирование новой популяции из особей-родителей и особей-потомков

№	Особи	Приспособленность	Новая популяция	Приспособленность особей в новой популяции
1	010	−0,33	001	−5,42
2	011	7,25	010	−0,33
3	101	7,92	110	5
4	100	10,33	011	7,25
5	001	−5,42		
6	110	5		
7	100	10,33		
8	111	12,58		

2.2. Основные понятия

Введем основные понятия, применяемые в генетических алгоритмах.

Вектор — упорядоченный набор чисел, называемых *компонентами* вектора[3]. Так как вектор можно представить в виде строки его координат, то в дальнейшем понятия вектора и строки считаются идентичными.

Булев вектор — вектор, компоненты которого принимают значения из двух элементного (булева) множества, например, $\{0, 1\}$ или $\{-1, 1\}$.

Хеммингово расстояние — используется для булевых векторов и равно числу различающихся в обоих векторах компонент.

Хеммингово пространство — пространство булевых векторов, с введенным на нем расстоянием (метрикой) Хемминга. В случае булевых векторов размерности n рассматриваемое пространство представляет собой множество вершин n-мерного гиперкуба с хеммингово метрикой. Расстояние между двумя вершинами определяется длиной кратчайшего соединяющего их пути, измеренной вдоль ребер.

Хромосома — вектор (или строка) из каких-либо чисел. Если этот вектор представлен бинарной строкой из нулей и единиц, например, 1010011, то он получен либо с использованием *двоичного кодирования*, либо *кода Грея* (см. *Приложение А.3*). Каждая позиция (бит) хромосомы называется *геном*.

Индивидуум (генетический код, особь) — набор хромосом (вариант решения задачи). Обычно особь состоит из одной хромосомы, поэтому в дальнейшем особь и хромосома идентичные понятия.

Расстояние — хеммингово расстояние между бинарными хромосомами.

Кро́ссинговер (кроссовер) — операция, при которой две хромосомы обмениваются своими частями. Например, 1100&1010 $\longrightarrow$ 1110&1000.

Мутация — случайное изменение одной или нескольких позиций в хромосоме. Например, 1010011 $\longrightarrow$ 1010001.

Инверсия — изменение порядка следования битов в хромосоме или в ее фрагменте. Например, 1100 $\longrightarrow$ 0011.

Популяция — совокупность индивидуумов.

Пригодность (приспособленность) — критерий или функция, экстремум которой следует найти.

Локус — позиция гена в хромосоме

Аллель — совокупность подряд идущих генов.

Эпистаз — влияние гена на пригодность индивидуума в зависимости от значения гена, присутствующего в другом месте. Ген считают эписта-

[3]Заметим, что данное определение вектора отличается от определения, принятого в математике.

тическим, когда его присутствие подавляет влияние гена в другом локусе. Эпистатические гены из-за их влияния на другие гены иногда называют ингибирующими. Подавление проявления гена неаллельным ему геном называется гипостазом, а сам подавляемый ген — гипостатическим.

Из приведенных выше определений следует, что терминология ГА представляет собой синтез собственно генетических и искусственных понятий. Так, для понятия, заимствованного из генетики, можно предъявить его искусственный (символический) аналог. Например, хромосома и строка. В биологических системах полный генетический пакет[4] называется *генотипом*. В искусственных системах полный генетический пакет строк называется *структурой*. В биологических системах в процессе индивидуального развития организма взаимодействие генотипа с окружающей средой формирует совокупность внешних признаков и свойств, называемую *фенотипом*. В математическом моделировании рассматриваемая структура декодируется с помощью *множества параметров*, которое в литературе иногда называют альтернативным решением или точкой. Всевозможные значения параметров образуют *пространство решений*. В искусственной генетической системе возможно использование как числовых, так и нечисловых параметров.

В биологической терминологии говорят, что хромосома образована генами. В генетике с любым локусом связана определенная генетическая функция. Поэтому можно говорить о специализированных генах. Например, ген цвета глаз животного находится в 10 локусах, т.е. голубой цвет глаз имеет 10-аллельное значение. В терминологии ГА говорят, что строки образованы значениями функции, или *детекторами*. Значения функции могут быть локализованы в различных позициях строки. Связь между естественной (биологической) и искусственной терминологией приведена в таблице 5.

Основные принципы работы ГА заключены в следующей схеме (см. также рис. 4):

1. Генерируем начальную популяцию из n хромосом.
2. Вычисляем для каждой хромосомы ее пригодность.
3. Выбираем пару хромосом-родителей с помощью одного из способов отбора.
4. Проводим *кроссинговер* двух родителей с вероятностью p_{c}, производя двух потомков.
5. Проводим мутацию потомков с вероятностью p_{m}.

[4] Совокупность всех генов.

6. Повторяем шаги 3–5, пока не будет сгенерировано новое поколение популяции, содержащее n хромосом.

7. Повторяем шаги 2–6, пока не будет достигнут критерий окончания процесса.

Таблица 5

Связь между естественной (биологической) и искусственной терминологией

Хромосома	Строка
Ген	Значение функции, характеристика, детектор
Аллель	Возможные значения генов
Локус	Позиция в строке
Генотип	Фактическая структура. Кодированная хромосома.
Фенотип	Множество параметров, альтернативное решение, декодированная структура
Эпистаз	Нелинейность. Взаимодействие генов.

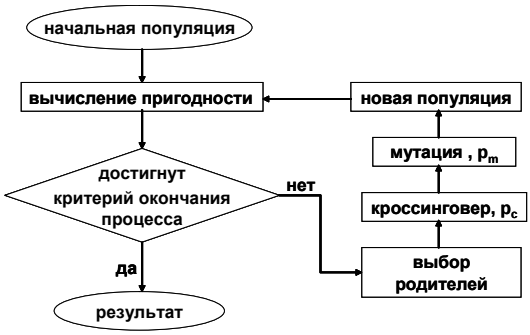

Рис. 4. Схема простого ГА

Критерием окончания процесса может служить заданное количество поколений или *схождение* (*convergence*) популяции.

Схождением называется такое состояние популяции, когда все строки популяции почти одинаковы и находятся в области некоторого экстремума. В такой ситуации кроссинговер практически никак не изменяет популяции, так как создаваемые при нем потомки представляют собой копии родителей с перемененными участками хромосом. Вышедшие из этой области

за счет мутации особи склонны вымирать, так как чаще имеют меньшую приспособленность, особенно если данный экстремум является глобальным максимумом. Таким образом, схождение популяции обычно означает, что найдено лучшее или близкое к нему решение.

Основными операторами ГА являются кроссинговер, мутация, выбор родителей и селекция (отбор хромосом в новую популяцию). Вид оператора играет важную роль в реализации и эффективности ГА. Существуют основные формы операторов, чистое использование или модернизация которых ведет к получению ГА, пригодного для решения конкретной задачи. Рассмотрим некоторые из них.

2.3. Операторы выбора родителей

Существует несколько подходов к выбору родительской пары. Наиболее распространенными *операторами выбора родителей* являются следующие.

Панмиксия — самый простой оператор отбора. В соответствии с ним каждому члену популяции сопоставляется случайное целое число на отрезке $[1; n]$, где n — количество особей в популяции. Будем рассматривать эти числа как номера особей, которые примут участие в скрещивании. При таком выборе какие-то из членов популяции не будут участвовать в процессе размножения, так как образуют пару сами с собой. Какие-то члены популяции примут участие в процессе воспроизводства неоднократно с различными особями популяции. Несмотря на простоту, такой подход универсален для решения различных классов задач. Однако он достаточно критичен к численности популяции, поскольку эффективность алгоритма, реализующего такой подход, снижается с ростом численности популяции.

Инбридинг представляет собой такой метод, когда первый родитель выбирается случайным образом, а вторым родителем является член популяции ближайший к первому. Здесь «ближайший» может пониматься, например, в смысле минимального расстояния Хемминга (для бинарных строк) или евклидова расстояния между двумя вещественными векторами. Расстояние Хемминга равно числу различающихся локусов (разрядов) в бинарной строке. Пример определения родства бинарных хромосом при выборе родительской пары для хромосомы 1010001 показан в табл. 6.

При *аутбридинге* также используют понятие схожести особей. Однако теперь брачные пары формируют из максимально далеких особей.

Последние два способа по разному влияют на поведение генетического алгоритма. Так, инбридинг можно охарактеризовать свойством концентрации поиска в локальных узлах, что фактически приводит к разбиению

популяции на отдельные локальные группы вокруг подозрительных на экстремум участков ландшафта. Аутбридинг же направлен на предупреждение сходимости алгоритма к уже найденным решениям, заставляя алгоритм просматривать новые, неисследованные области. Инбридинг и аутбридинг бывает генотипным (когда в качестве расстояния берется разность значений целевой функции для соответствующих особей) и фенотипным (в качестве расстояния берется расстояние Хемминга).

Таблица 6

Хеммингово расстояние между хромосомами популяции и хромосомой 1010001

Хромосомы популяции	Количество отличающихся локусов
1000000	2
1010101	1
1111111	4
1100001	2
0110011	3
0100011	4
0011100	4
0000000	3

Селекция состоит в том, что родителями могут стать только те особи, значение приспособленности которых не меньше пороговой величины, например, среднего значения приспособленности по популяции. Такой подход обеспечивает более быструю сходимость алгоритма. Однако из-за быстрой сходимости селективный выбор родительской пары не подходит тогда, когда ставится задача определения нескольких экстремумов, поскольку для таких задач алгоритм, как правило, быстро сходится к одному из решений. Кроме того, для некоторых многомерных задач со сложным ландшафтом целевой функции быстрая сходимость может превратиться в преждевременную сходимость к квазиоптимальному решению. Этот недостаток может быть отчасти компенсирован использованием подходящего механизма отбора, который бы «тормозил» слишком быструю сходимость алгоритма.

Пороговая величина в селекции может быть вычислена разными способами. Поэтому в литературе по ГА выделяют различные вариации селекции. Наиболее известные из них — это *турнирный* и *рулеточный* (пропорциональный) отборы.

При *турнирном отборе (tournament selection)* из популяции, содержащей N особей, выбираются случайным образом t особей, и лучшая из

них особь записывается в промежуточный массив (рис. 5). Эта операция повторяется N раз. Особи в полученном промежуточном массиве затем используются для скрещивания (также случайным образом). Размер группы строк, отбираемых для турнира, часто равен 2. В этом случае говорят о двоичном (парном) турнире. Вообще же t называют численностью турнира. Преимуществом данного способа является то, что он не требует дополнительных вычислений.

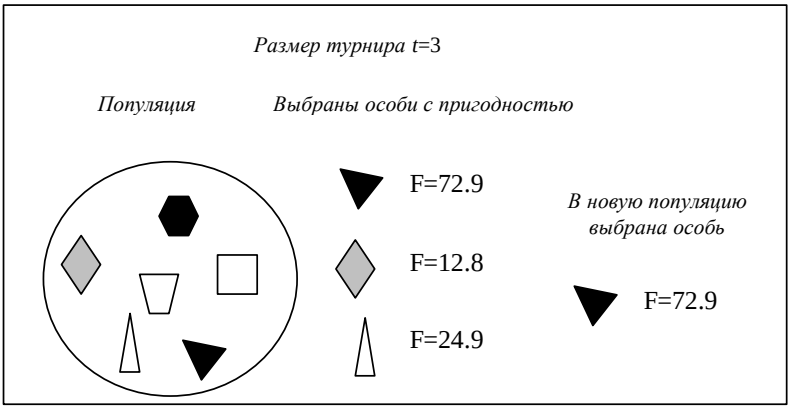

Рис. 5. Турнирный отбор

В *методе рулетки (roulette-wheel selection)* особи отбираются с помощью N «запусков» рулетки, где N — размер популяции. Колесо рулетки содержит по одному сектору для каждого члена популяции. Размер i-го сектора пропорционален вероятности попадания в новую популяцию $P(i)$, вычисляемой по формуле:

$$P(i) = \frac{f(i)}{\sum\limits_{i=1}^{N} f(i)},$$

где $f(i)$ — пригодность i-й особи. Ожидаемое число копий i-ой хромосомы после оператора рулетки определяются по формуле $N_i = P(i)N$.

При таком отборе члены популяции с более высокой приспособленностью с большей вероятностью будут чаще выбираться, чем особи с низкой приспособленностью (табл. 7).

Другие способы отбора можно получить на основе модификации выше

приведенных. Так, например, в рулеточном отборе можно изменить формулу для вероятности попадания особи в новую популяцию.

2.4. Рекомбинация (воспроизведение)

Оператор рекомбинации применяют сразу же после оператора отбора родителей для получения новых особей-потомков. Смысл рекомбинации заключается в том, что созданные потомки должны наследовать генную информацию от обоих родителей. Различают *дискретную рекомбинацию* и *кроссинговер*.

2.4.1. Дискретная рекомбинация

Дискретная рекомбинация *(Discrete recombination)* в основном применяется к хромосомам с вещественными генами. Основными способами дискретной рекомбинации являются собственно дискретная рекомбинация, промежуточная, линейная и расширенно линейная рекомбинации.

Дискретная рекомбинация соответствует обмену генами между особями. Для иллюстрации данного оператора сравним две особи с тремя генами:

Особь 1	12	25	7
Особь 2	116	4	34

Для создания двух потомков с равной вероятностью случайно выберем номер особи для каждого гена:

Схема 1	2	2	1
Схема 2	1	2	1

Таблица 7

Метод рулетки. Суммарная пригодность $= 200$, суммарная вероятность $= 1$

Популяция из 5 особей	Пригодность	Вероятность выбора
C_1	52	$52/200 = 0{,}26$
C_2	85	$85/200 = 0{,}425$
C_3	37	$37/200 = 0{,}185$
C_4	3	$3/200 = 0{,}015$
C_5	23	$23/200 = 0{,}115$

Согласно схеме создадим потомков:

Потомок 1	116	4	7
Потомок 2	12	4	7

Дискретная рекомбинация применима для любого типа генов (двоичные, вещественные и символьные).

Промежуточная рекомбинация *(Intermediate recombination)* применима только к вещественным переменным, но не к бинарным. В данном методе предварительно определяется числовой интервал значений генов потомков, который должен содержать значения генов родителей. Потомки создаются по следующему правилу:

$$\text{Потомок} = \text{Родитель } 1 + \alpha\cdot(\text{Родитель } 2 - \text{Родитель } 1),$$

где множитель α — случайное число на отрезке $[-d, 1+d]$, $d \geqslant 0$. Как отмечают сторонники этого метода, наиболее оптимальное воспроизведение получается при $d = 0{,}25$. Для каждого гена создаваемого потомка выбирается отдельный множитель α. Рассмотрим применение оператора на примере. Пусть два родителя имеют следующие значения генов:

Особь 1	12	25	7
Особь 2	116	4	34

Случайно выберем значения $\alpha \in [-0{,}25; 1{,}25]$ для каждого гена обоих потомков:

Схема 1	0,5	1,1	$-0{,}1$
Схема 2	0,1	0,8	0,5

Вычислим значения генов потомков по предложенной выше формуле:

Потомок 1	$12 + 0{,}5(116 - 12) = 64$	$25 + 1{,}1(4 - 25) = 1{,}9$	4,3
Потомок 2	$12 + 0{,}1(116 - 12) = 22{,}4$	$25 + 0{,}8(4 - 25) = 8{,}2$	20,5

При промежуточной рекомбинации возникают значения генов, отличные от значения генов особей-родителей. Это приводит к возникновению новых особей, пригодность которых может быть лучше, чем пригодность родителей. В литературе такой оператор рекомбинации иногда называется *дифференциальным скрещиванием*.

Линейная рекомбинация *(Line recombination)* отличается от промежуточной тем, что множитель α выбирается для каждого потомка один раз. Рассмотрим гены приведенных выше родителей. Пусть значение α определяется следующим образом:

Схема 1	0,5
Схема 2	0,1

Тогда гены созданных потомков примут следующие значения:

Потомок 1	$12 + 0{,}5(116 - 12) = 64$	$25 + 0{,}5(4 - 25) = 14{,}5$	20,5
Потомок 2	$12 + 0{,}1(116 - 12) = 22{,}4$	$25 + 0{,}1(4 - 25) = 22{,}9$	9,7

Если рассматривать особи популяции как точки в k-мерном пространстве, где k — количество генов в одной особи, то можно сказать, что при линейной рекомбинации генерируемые точки потомков лежат на прямой, заданной двумя точками — родителями.

2.4.2. Кроссинговер (бинарная рекомбинация)

Рекомбинацию бинарных строк принято называть кроссинговером (кроссовером) или скрещиванием.

Одноточечный кроссинговер *(Single-point crossover)* моделируется следующим образом. Пусть имеются две родительские особи с хромосомами $X = \{x_i, i \in [0; L]\}$ и $Y = \{y_i, i \in [0; L]\}$. Случайным образом определяется точка внутри хромосомы (точка разрыва), в которой обе хромосомы делятся на две части и обмениваются ими. Такой тип кроссинговера называется *одноточечным*, так как при нем родительские хромосомы разделяются только в одной случайной точке.

Родители		Потомки
$x_1x_2x_3 \dots x_{n-1}x_n \dots x_m$	$\rightarrow$	$x_1x_2x_3 \dots x_{n-1}y_n \dots y_m$
$y_1y_2y_3 \dots y_{n-1}y_n \dots y_m$	$\rightarrow$	$y_1y_2y_3 \dots y_{n-1}x_n \dots x_m$

Рис. 6. Одноточечный кроссинговер

Также применяется двух- и N-точечный кроссинговер.

В **двухточечном кроссинговере** (и многоточечном кроссинговере вообще) хромосомы рассматриваются как циклы, которые формируются соединением концов линейной хромосомы вместе. Для замены сегмента одного цикла сегментом другого цикла требуется выбор двух точек разреза. В этом представлении, одноточечный кроссинговер может быть рассмотрен как кроссинговер с двумя точками, но с одной точкой разреза, зафиксированной в начале строки. Следовательно, двухточечный кроссинговер решает ту же самую задачу, что и одноточечный, но более полно. Хромосома, рассматриваемая как цикл, может содержать большее количество стандартных блоков, так как они могут совершить «циклический возврат» в конце

строки (рис. 7). В настоящий момент многие исследователи соглашаются, что двухточечный кроссинговер вообще лучше, чем одноточечный.

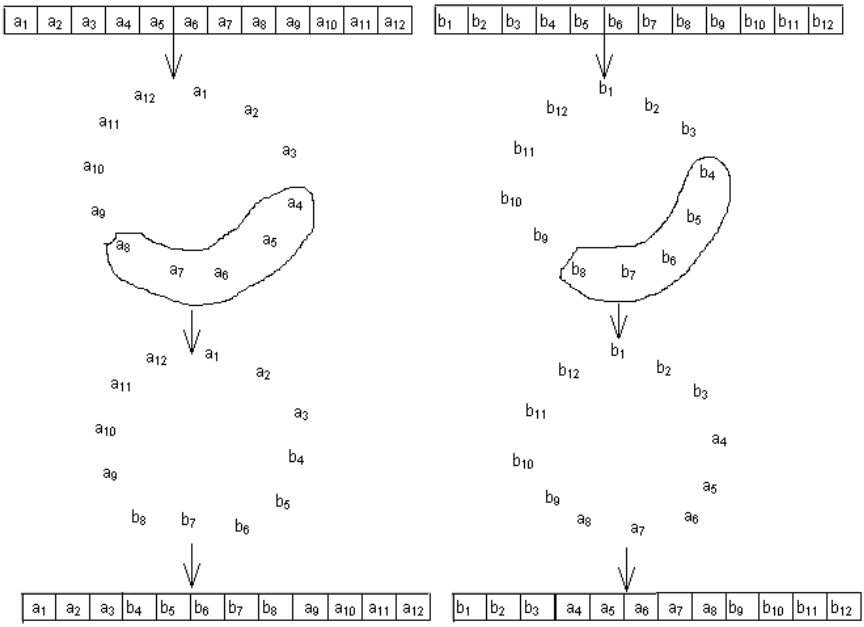

Рис. 7. Двухточечный кроссинговер

Для **многоточечного кроссинговера** *(Multi–point crossover)*, выбираем m точек разреза $k_i \in \{1, 2, \ldots, Nvar\}$, $i = 1 : m$, $Nvar$ — количество переменных (генов) в особи. Точки разреза выбираются случайно без повторений и сортируются в порядке возрастания. При кроссинговере происходит обмен участками хромосом, ограниченными точками разреза и таким образом получают двух потомков. Участок особи с первым геном до первой точки разреза в обмене не участвует. Сравним следующие две особи по 11 двоичным генам.

Особь 1	0	1	1	1	0	0	1	1	0	1	0
Особь 2	1	0	1	0	1	1	0	0	1	0	1

Выберем точки разреза кроссинговера:

Точка разреза ($m = 3$)	2	6	10

Создадим двух новых потомков:

Потомок 1	0	1	1	0	1	1	1	1	0	1	1
Потомок 2	1	0	1	1	0	0	0	0	1	0	0

Применение многоточечного кроссинговера требует введения нескольких переменных (точек разреза), и для воспроизведения выбираются особи с наибольшей приспособленностью.

Одноточечный и многоточечный кроссинговер определяют точки разреза, которые разделяют особи на части. **Однородный кроссинговер** *(Uniform crossover)* создает маску (схему) особи, в каждом локусе которой находится потенциальная точка кроссинговера. Маска кроссинговера имеет ту же длину, что и скрещивающиеся особи. Создать маску можно следующим образом: введем некоторую величину $0 < p_0 < 1$, и если случайное число больше p_0, то на n-ю позицию маски ставим 0, иначе — 1. Таким образом, гены маски представляют собой случайные двоичные числа (0 или 1). Согласно этим значениям, геном потомка становится первая (если ген маски $= 0$) или вторая (если ген маски $= 1$) особь-родитель.

Например, рассмотрим особи:

Особь 1	0	1	1	1	0	0	1	1	0	1	0
Особь 2	1	0	1	0	1	1	0	0	1	0	1

Для каждого создаваемого потомка создадим маску из 11 случайно выбранных элементов из множества $\{0; 1\}$:

Маска 1	0	1	1	0	0	0	1	1	0	1	0
Маска 2	1	0	0	1	1	1	0	0	1	0	1

Создадим потомков по следующему правилу: если на i-ом месте в соответствующей маске стоит 1, то ген 1 родителя переходит потомку, иначе наследуется ген второго родителя. Получим следующие особи:

Потомок 1	1	1	0	1	1	1	1	1	1	1	1
Потомок 2	0	0	1	1	0	0	0	0	0	0	0

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

Алгоритм однородного кроссинговера для двоичных строк полностью идентичен дискретному воспроизведению для вещественных хромосом. Такой вид кроссинговера еще называют унифицированным.

Триадный кроссинговер *(Triadic crossover)*. Данная разновидность кроссинговера отличается от однородного тем, что после отбора пары родителей из остальных членов популяции случайным образом выбирается

особь, которая в дальнейшем используется в качестве маски. Далее 10 % генов маски мутируют. Затем гены первого родителя сравниваются с генами маски: если гены одинаковы, то они передаются первому потомку, в противном случае на соответствующие позиции хромосомы потомка переходят гены второго родителя. Генотип второго потомка отличается от генотипа первого тем, что на тех позициях, где у первого потомка стоят гены первого родителя, у второго потомка стоят гены второго родителя и наоборот.

Перетасовочный кроссинговер *(Shuffler crossover)*. В данном алгоритме особи, отобранные для кроссинговера, случайным образом обмениваются генами. Затем выбирают точку для одноточечного кроссинговера и проводят обмен частями хромосом. После скрещивания созданные потомки вновь тасуются. Таким образом, при каждом кроссинговере создаются не только новые потомки, но и модифицируются родители (старые родители удаляются), что позволяет сократить число операций по сравнению с однородным кроссинговером.

Кроссинговер с уменьшением замены *(Crossover with reduced surrogate)*. Оператор уменьшения замены ограничивает кроссинговер, чтобы всегда, когда это возможно, создавать новые особи. Это осуществляется за счет ограничения на выбор точки разреза: точки разреза должны появляться только там, где гены различаются.

Как было показано выше, кроссинговер генерирует новое решение (в виде особи-потомка) на основе двух имеющихся, комбинируя их части. Поэтому число различных решений, которые могут быть получены кроссинговером при использовании одной и той же пары готовых решений, ограничено. Соответственно, пространство, которое ГА может покрыть, используя лишь кроссинговер, жестко зависит от генофонда популяции. Чем разнообразнее генотипы решений популяции, тем больше пространство покрытия. При обнаружении локального оптимума соответствующий ему генотип будет стремиться занять все позиции в популяции, и алгоритм может сойтись к ложному оптимуму. Поэтому в генетическом алгоритме важную роль играют мутации. Существует несколько способов мутирования генов. Вопрос о выборе подходящего оператора мутации решается в рамках поставленной задачи.

2.5. Мутация

После процесса воспроизводства происходят мутации *(mutation)*. Данный оператор необходим для «выбивания» популяции из локального экс-

тремума и препятствует преждевременной сходимости. Это достигается за счет того, что изменяется случайно выбранный ген в хромосоме (рис. 8).

$$x_1x_2\ldots x_{n-1}x_nx_{n+1}\ldots x_m \longrightarrow x_1x_2\ldots x_{n-1}\bar{x}_nx_{n+1}\ldots x_m$$

Рис. 8. Мутация в точке x_n

Так же как и кроссинговер, мутации могут проводиться не только по одной случайной точке. Можно выбирать для изменения несколько точек в хромосоме, причем их число также может быть случайным. Используют и мутации с изменением сразу некоторой группы подряд идущих точек.

Вероятность мутации p_{m} (как правило, $p_{\text{m}} \ll 1$) может являться или фиксированным случайным числом на отрезке $[0;1]$, или функцией от какой-либо характеристики решаемой задачи. Например, можно положить вероятность мутирования генов, обратно пропорциональную числу всех генов в особи (размерности).

Оптимальное значение вероятности мутации обсуждается в разных статьях. Так, например, мутация с фиксированной вероятностью приводит к хорошим результатам для широкого класса тестовых функций (например, для унимодальных функций[5]). Для мультимодальных функций применяют самоадаптирующуюся оценку вероятности. Ниже приведены основные варианты мутирования.

Мутация для вещественных особей *(Real valued mutation)*. На рис. 9 показана возможная мутация для особей с вещественными генами в двумерном пространстве.

Для мутации особей с вещественными числами необходимо определить величину шага мутации — число, на которое изменится значение гена при мутировании.

Обычно определение шага мутации представляет некоторую трудность. Оптимальный размер шага должен меняться в течение всего процесса поиска. Наиболее пригодны маленькие шаги, но иногда большие шаги могут привести к ускорению процесса. Гены могут мутировать согласно следующему правилу:

$$\text{новая переменная} = \text{старая переменная} \pm \alpha \cdot \delta,$$

[5]Унимодальная функция — одновершинная функция.

где знаки $+$ или $-$ выбираются с равной вероятностью, $\alpha = 0{,}5 \times$ $\times$поисковое пространство (интервал изменения данной переменной),

$$\delta = \sum_{i=1}^{m} a(i)2^{-i},$$

$a(i) = 1$ с вероятностью $\frac{1}{m}$, в противном случае $a(i) = 0$, m — параметр.

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

Двоичная мутация *(Binary mutation)*. Для особей, кодированных двоичным кодом или кодом Грея, мутация заключается в случайном инвертировании гена (0 заменяется 1 и наоборот). Эффект мутации зависит от примененного способа кодирования генов. Так, в одних задачах при мутации наилучший эффект достигается в случае, когда особи закодированы кодом Грея, а в других — с помощью двоичного кода.

Плотность мутации *(Density mutation)*. Стратегия мутации с использованием понятия плотности заключается в мутировании каждого гена потомка с заданной вероятностью. Таким образом, кроме вероятности применения мутации к самому потомку используется еще вероятность применения мутации к каждому его гену, величину которой выбирают с таким расчетом, чтобы в среднем мутировало от 1 до 10 % генов.

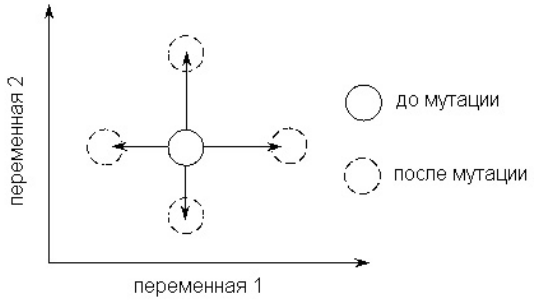

Рис. 9. Мутация для особей с вещественными генами

Другие виды мутаций. Пусть особь t представлена следующей последовательностью генов t_i: $t = t_1, \ldots, t_k$. Тогда можно применить следующие операторы мутации:

1. *Присоединение* случайного гена из совокупности всевозможных значений генов к концу последовательности: $t \to t_1, \ldots, t_k, s$.

2. *Вставка* случайного гена из совокупности всевозможных значений генов в случайно выбранную позицию в последовательности: $t \to t_1, \ldots, t_{i-1}, s, t_i, \ldots, t_k$.

3. *Удаление* случайно выбранного гена из последовательности: $t \to t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k$.

4. *Обмен* местами в последовательности двух соседей одного случайно выбранного гена: $t \to t_1, \ldots, t_{i+1}, s, t_{i-1}, \ldots, t_k$.

Следует заметить, что мутация 1 является частным случаем мутации 2. Для особей с фиксированным размером (количество генов в последовательности) возможно применение в чистом виде только 4 мутации, а 1 и 2 мутации должны применяться в сочетании с мутацией 3.

2.6. Операторы отбора особей в новую популяцию

Для создания новой популяции можно использовать различные методы отбора особей.

Отбор усечением *(Truncation selection)*. При отборе усечением используют популяцию, состоящую как из особей-родителей, так и особей-потомков, отсортированную по возрастанию значений функции пригодности особей. Число особей для скрещивания выбирается в соответствии с порогом $T \in [0; 1]$. Порог определяет, какая доля особей, начиная с самой первой (самой пригодной), будет принимать участие в отборе. В принципе, порог можно задать и числом больше 1, тогда он будет просто равен числу особей из текущей популяции, допущенных к отбору. Среди особей, попавших «под порог», случайным образом выбирается одна и записывается в новую популяцию. Процесс повторяется N раз, пока размер новой популяции не станет равен размеру исходной популяции. Новая популяция состоит только из особей с высокой пригодностью, причем одна и та же особь может встречаться несколько раз, а некоторые особи, имеющие пригодность выше пороговой, могут не попасть в новую популяцию.

Из-за того, что в этой стратегии используется отсортированная популяция, время ее работы может быть большим для популяции большого размера и зависеть также от алгоритма сортировки.

Элитарный отбор *(Elite selection)*. Создается промежуточная популяция, которая включает в себя как родителей, так и их потомков. Члены этой популяции оцениваются, а за тем из них выбираются N самых лучших (пригодных), которые и войдут в следующее поколение. Зачастую данный метод комбинируют с другими — выбирают в новую популяцию, например, 10 % «элитных» особей, а остальные 90 % — одним из традиционных методов селекции. Иногда эти 90 % особей создают случайно, как при создании начальной популяции перед запуском работы генетического алгоритма. Использование стратегии элитизма оказывается весьма полезным для эффективности ГА, так как не допускает потерю лучших решений. К примеру, если популяция сошлась в локальном максимуме, а мутация вывела одну из строк в область глобального, то при предыдущей стратегии весьма вероятно, что эта особь в результате скрещивания будет потеряна, и решение задачи не будет получено. Если же используется элитизм, то полученное хорошее решение будет оставаться в популяции до тех пор, пока не будет найдено еще лучшее.

Отбор вытеснением *(Exclusion selection)*. В данном отборе выбор особи в новую популяцию зависит не только от величины ее пригодности, но и от того, есть ли уже в формируемой популяции особь с аналогичным хромосомным набором. Отбор проводится из числа родителей и их потомков. Из всех особей с одинаковой приспособленностью предпочтение сначала отдается особям с разными генотипами. Таким образом, достигаются две цели: во-первых, не теряются лучшие найденные решения, обладающие различными хромосомными наборами, во-вторых, в популяции постоянно поддерживается генетическое разнообразие. Вытеснение в данном случае формирует новую популяцию скорее из удаленных особей, вместо особей, группирующихся около текущего найденного решения. Данный метод наиболее пригоден для многоэкстремальных задач, при этом помимо определения глобальных экстремумов появляется возможность выделить и те локальные максимумы, значения которых близки к глобальным.

Метод Больцмана, или метод отжига *(Bolzman selection)*. В данном методе вероятность отбора в новую популяцию зависит от управляющего параметра — температуры T.

Обычно вероятность попадания в новую популяцию вычисляется по следующей формуле:

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

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

В некоторых случаях применяется альтернативная формула:

$$p = \frac{\exp(f(i)/T)}{\langle \exp(f(i)/T) \rangle},$$

где $\langle\rangle$ — среднее по популяции на итерации с номером t. Если p окажется больше случайного числа на интервале $(0;1)$, то особь $f(i)$ попадет в новую популяцию.

В данном методе первым поколениям соответствуют высокие температуры, и вероятность отбора особей велика (поддерживается многообразие в новой популяции). Постепенно с ростом количества поколений ГА температура снижается, вероятность отбора уменьшается и в новую популяцию попадают те особи, приспособленность которых минимальна. Данный метод отбора используется для узкого класса задач (например, нахождение основного состояния спиновых стекол).

2.7. Разнообразие генетических алгоритмов

Следует отметить, что в настоящее время ГА — это целый класс алгоритмов, направленный на решение разнообразных задач. Примерами различных ГА могут являться следующие алгоритмы.

2.7.1. Канонический ГА

Данная модель алгоритма является классической. Она была предложена Джоном Холландом в его работе [8]. Согласно ей, популяция состоит из N хромосом с фиксированной разрядностью генов. С помощью пропорционального отбора формируется промежуточный массив, из которого случайным образом выбираются два родителя. Далее производится одноточечный кроссинговер, и созданные два потомка мутируют (одноточечная мутация) с заданной вероятностью. Мутировавшие потомки занимают места своих родителей. Процесс продолжается до тех пор, пока не будет достигнут критерий окончания алгоритма.

2.7.2. Генитор

В модели генитор (*Genitor*) используется специфичный способ отбора [4]. Вначале, как и полагается, популяция инициализируется, и ее особи оцениваются. Затем выбираются случайным образом две особи, скрещиваются, причем получается только один потомок, который оценивается и занимает место менее приспособленной особи в популяции (а не одного из родителей!). После этого снова случайным образом выбираются две особи, и их потомок занимает место родительской особи с самой низкой приспособленностью. Таким образом, на каждом шаге в популяции обновляется лишь одна особь. Процесс продолжается до тех пор, пока пригодности хромосом не станут одинаковыми. В данный алгоритм можно добавить мутацию потомка после его создания. Критерий окончания процесса, как и вид кроссинговера и мутации, можно выбирать разными способами.

2.7.3. Метод прерывистого равновесия

Данный метод основан на палеонтологической теории прерывистого равновесия, которая описывает быструю эволюцию за счет вулканических и других изменений земной коры. Для применения данного метода в технических задачах предлагается после каждой генерации промежуточного поколения случайным образом перемешивать особи в популяции, а затем применять основной ГА. В данной модели для отбора родительских пар используется панмиксия. Получившиеся в результате кроссинговера потомки и наиболее пригодные родители случайным образом смешиваются. Из общей массы в новое поколения попадут лишь те особи, пригодность которых выше средней. Тем самым достигается управление размером популяции в зависимости от наличия лучших особей. Такая модификация метода прерывистого равновесия может позволить сократить неперспективные популяции и расширить популяции, в которых находятся лучшие индивидуальности. Как пишет В.В Курейчик: «метод прерывистого равновесия — это мощный стрессовый метод изменения окружающей среды, который используется для эффективного выхода из локальных ям» [?].

2.7.4. Гибридный алгоритм

Идея **гибридных алгоритмов** *(Hybrid algorithms)* заключается в сочетании генетического алгоритма с некоторым другим классическим методом поиска, подходящим в данной задаче. В каждом поколении все сгенерированные потомки оптимизируются выбранным методом и затем заносятся

в новую популяцию. Тем самым получается, что каждая особь в популяции достигает локального оптимума, вблизи которого она находится. Далее производятся обычные для ГА действия: отбор родительских пар, кроссинговер и мутации. На практике гибридные алгоритмы оказываются очень удачными. Это связано с тем, что вероятность попадания одной из особей в область глобального максимума обычно велика. После оптимизации такая особь будет являться решением задачи.

Известно, что генетический алгоритм способен быстро найти во всей области поиска хорошие решения, но он может испытывать трудности в получении из них наилучших. Обычный оптимизационный метод может быстро достичь локального максимума, но не может найти глобальный. Сочетание двух алгоритмов позволяет использовать преимущества обоих.

2.7.5. CHC

CHC (Eshelman, 1991) расшифровывается как *Cross-population selection, Heterogeneous recombination and Cataclysmic mutation*. Данный алгоритм довольно быстро сходится из-за того, что в нем нет мутаций, следующих за оператором кроссинговера, используются популяции небольшого размера, и отбор особей в следующее поколение ведется и между родительскими особями, и между их потомками. В данном методе для кроссинговера выбирается случайная пара, но не допускается, чтобы между родителями было маленькое хеммингово расстояние или мало расстояние между крайними различающимися битами. Для скрещивания используется разновидность однородного кроссовера *HUX* (*Half Uniform Crossover*), при котором потомку переходит ровно половина битов каждого родителя. Для нового поколения выбираются N лучших различных особей среди родителей и детей. При этом дублирование строк не допускается. В модели CHC размер популяции относительно мал — около 50 особей. Это оправдывает использование однородного кроссинговера и позволяет алгоритму сойтись к решению. Для получения более или менее одинаковых строк (см. определение сходимости) CHC применяет *cataclysmic mutation*: все строки, кроме самой приспособленной, подвергаются сильной мутации (изменяется около трети битов). Таким образом, алгоритм перезапускается и далее продолжает работу, применяя только кроссинговер.

2.7.6. ГА с нефиксированным размером популяции

В генетическом алгоритме с нефиксированным размером популяции (Genetic Algorithm with Varying Population Size — GAVaPS) каждой особи

приписывается максимальный возраст, т.е. число поколений, по прошествии которых особь погибает. Внедрение в алгоритм нового параметра — возраста — позволяет исключить оператор отбора в новую популяцию. Возраст каждой особи индивидуален и зависит от ее приспособленности. В каждом поколении t на этапе воспроизведения обычным образом создается дополнительная популяция из потомков. Размер дополнительной популяции ($AuxPopsize(t)$) пропорционален размеру основной популяции ($Popsize(t)$) и равен

$$AuxPopsize(t) = [Popsize(t)p_c],$$

где p_c — вероятность воспроизведения. Для воспроизведения особи выбираются из основной популяции с равной вероятностью независимо от их приспособленности. После применения мутации и кроссинговера потомкам приписывается возраст согласно значению их приспособленности. Возраст является константой на протяжении всей эволюции особи (от рождения до гибели). Затем из основной популяции удаляются те особи, срок жизни которых истек, и добавляются потомки из промежуточной популяции. Таким образом, размер после одной итерации алгоритма вычисляется по формуле:

$$Popsize(t+1) = Popsize(t) + AuxPopsize(t) - D(t),$$

где $D(t)$ — число особей, которые умирают в поколении t.

2.8. Параллельное выполнение ГА

Генетические алгоритмы применяются и при параллельных вычислениях (Parallel implementations). При этом формируется несколько живущих отдельно популяций. На этапе формирования нового поколения по некоторому правилу отбираются особи из разных популяций. Сгенерированная таким образом популяция в некоторых случаях заменяет все остальные. Таким образом, так или иначе, происходит миграция особей одной популяции в другие популяции. Поэтому параллельные ГА называют миграциями.

2.8.1. Параллельный ГА

Для начала рассмотрим создание простейшего параллельного ГА из классической модели Холланда. Для этого будем использовать турнирный отбор. Заведем $N/2$ процессов (здесь и далее процесс рассматривается как некоторая машина, процессор, который может работать независимо). Каждый из них будет выбирать случайно из популяции 4 особи, проводить 2

турнира, и победителей скрещивать. Полученные потомки будут записываться в новое поколение. Таким образом, за один цикл работы одного процесса будет сменяться целое поколение.

2.8.2. Миграция

Модель миграции *(Migration)* представляет популяцию как множество подпопуляций. Каждая подпопуляция обрабатывается отдельным процессором. Эти подпопуляции развиваются независимо друг от друга в течение одинакового количества поколений T (время изоляции). По истечении времени изоляции происходит обмен особями между подпопуляциями (миграция). Количество особей, подвергшихся обмену (вероятность миграции), метод отбора особей для миграции и схема миграции определяет частоту возникновения генетического многообразия в подпопуляциях и обмен информацией между популяциями.

Отбор особей для миграции может происходить следующим образом:

- случайная однообразная выборка из числа особей;
- пропорциональный отбор: для миграции берутся наиболее пригодные особи.

Отдельные подпопуляции в параллельных ГА можно условно принять за вершины некоторого графа. В связи с этим можно рассматривать топологию графа миграционного ГА. Наиболее распространенной топологией миграции является полный граф (см. рис. 10), при которой особи из любой подпопуляций могут мигрировать в любую другую подпопуляцию. Для каждой подпопуляции полное количество потенциальных иммигрантов строится на основе всех подпопуляций. Мигрирующая особь случайным образом выбирается из этого общего числа.

При использовании в неограниченной миграции пропорционального отбора сначала формируется массив из наиболее пригодных особей, отобранных по всем подпопуляциям. Случайным образом из этого массива выбирается особь, и ею заменяют наименее пригодную особь в подпопуляции 1. Аналогичные действия проделываем с остальными подпопуляциями. Возможно, что какая-то популяция получит дубликат своей «хорошей» особи.

Другая основная миграционная схема — это топология кольца (см. рис. 11). Здесь особи передаются между соседними (по направлению обхода) подпопуляциями. Таким образом, особи из одной подпопуляции могут мигрировать только в одну — соседнюю подпопуляцию.

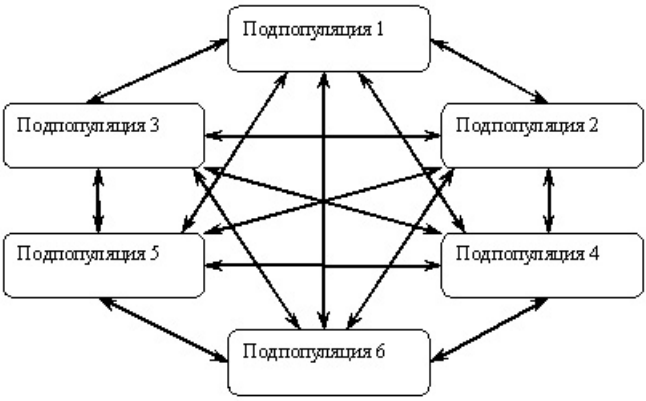

Рис. 10. Миграция с топологией полной сети

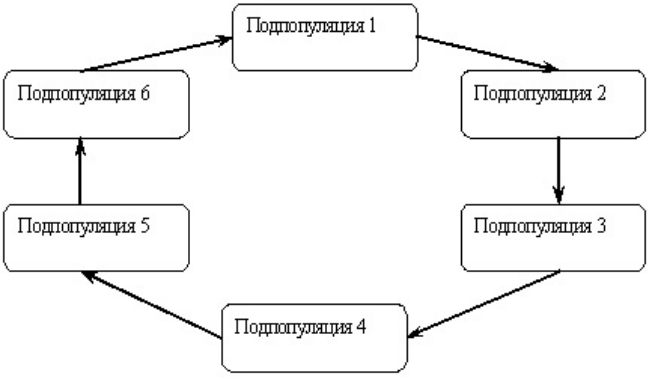

Рис. 11. Миграция с топологией кольца

На рис. 12 представлена стратегия миграции, похожая на топологию кольца. Как и при топологии кольца, миграция осуществляется только между ближайшими соседями. Однако миграция в этой модели также возможна между «крайними» подпопуляциями (тороидальные краевые миграции).

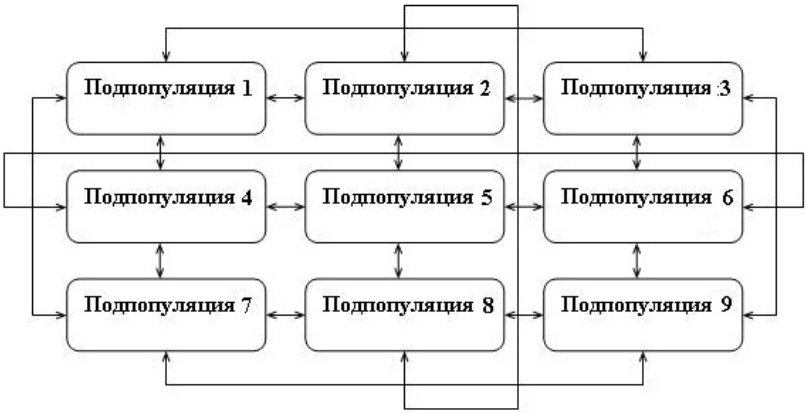

Рис. 12. Миграция с модифицированной топологией кольца

2.8.3. Глобальная модель «Рабочий и Хозяин»

Алгоритм «Рабочий и Хозяин» (Global model — worker/farmer) реализуется с применением нескольких одновременно работающих компьютеров (см. рис. 13). Среди всех компьютеров выделяют «Хозяина» и «Рабочих». Компьютеры — «Рабочие» отвечают за процессы воспроизводства, мутации и вычисления функции пригодности особей для их отбора в новое поколение. Все созданные и оцененные «Рабочими» особи поступают к компьютеру — «Хозяину», который затем проводит отбор особей согласно оценке их пригодности в новую популяцию. Отобранные особи передаются «Хозяином» к компьютерам — «Рабочим».

Рис. 13. Модель «Рабочий и Хозяин»

2.8.4. Модель диффузии, или островная модель ГА

Островная модель является наиболее распространенной моделью параллельного ГА. Ее суть заключается в том, что популяция, как правило состоящая из очень большого числа особей, разбивается на одинаковые по размеру подпопуляции. Каждая подпопуляция обрабатывается отдельным процессором с помощью одной из разновидностей непараллельного ГА. Изредка, например, через пять поколений, подпопуляции будут обмениваться несколькими особями. Такие миграции позволяют подпопуляциям совместно использовать генетический материал.

Пусть выполняются 16 независимых генетических алгоритмов, используя подпопуляции из 1000 особей на каждом процессоре. Если миграций нет, то происходит 16 независимых поисков решения. Все поиски ведутся на различных начальных популяциях и сходятся к определенным особям. Исследования подтверждают, что генетический дрейф склонен приводить подпопуляции к различным доминирующим особям. Это связано с тем, что, во-первых, количество островов, принимающих доминирующих «эмигрантов» с острова, ограничено (2–5 островов). Во-вторых, обмен особями односторонен. Поэтому в большой популяции появятся группы островов с различными доминирующими особями. Если популяция небольшого размера, то возможно быстрое мигрирование ложных доминирующих особей. Например, истинное решение находится только на одном острове, а несколько ложных доминант — на других островах. Тогда при миграции количе-

ство ложных особей на островах возрастет (на каждый остров мигрирации происходят с не менеее 2 островов), генетическим алгоритмом верное решение будет разрушено. Тем самым в маленькой популяции при генетическом дрейфе возможно появление ошибочных доминирующих особей и схождение алгоритма к ложному оптимуму.

Введение миграций в островной модели позволяет находить различные особи-доминанты в подпопуляциях, что способствует поддержанию многообразия в популяции. Каждую подпопуляции можно принять за остров. Во время миграции подпопуляции обмениваются своим доминирующим генетическим материалом. При частом мигрировании большого количества особей происходит перемешивание генетического материала. Тем самым устраняются локальные различия между островами. Очень редкие миграции не позволяют предотвратить преждевременную сходимость алгоритма в маленьких подпопуляциях. В рассматриваемой модели с каждого острова миграции могут происходить лишь на определенное расстояние: 2–5 острова в зависимости от количества подпопуляций. Таким образом, каждый остров оказывается почти изолированным. Количество островов, на которые могут мигрировать особи одной подпопуляции, называют расстоянием изоляции. Следует заметить, что взаимомиграции исключены (рис. 14).

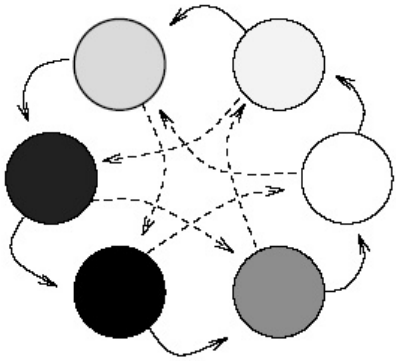

Рис. 14. Островная модель

Все приведенные варианты ГА можно моделировать, задавая разные значения параметров ГА и типы операторов. Например, в островной модели ГА можно моделировать различные комбинации способов отбора и формирования следующего поколения или сделать так, что в разных популяциях будут использоваться разные комбинации операторов ГА.

3. Параметры ГА

Вероятность кроссинговера обычно выбирается достаточной высокой 80–95 %. Однако, в некоторых задачах наилучший результат достигается при кроссинговере с вероятностью 60 %.

Вероятность мутации должна быть мала: 0,5–1 %.

Как это ни странно, очень большой размер популяции обычно не приводит к хорошим результатам (скорость сходимости алгоритма не увеличивается). Оптимальный размер популяции — 20–30 особей, однако в некоторых задачах, как пишут исследователи, требуется 50–100 особей. Исследования показывают, что оптимальный размер популяции зависит от размера кодовых строк (хромосом). Так, для алгоритма с 32-битовыми хромосомами размер популяции будет больше, чем для алгоритма с 16-битовыми хромосомами.

В основном используют рулеточный отбор, но и иногда лучше применить отбор с усечением. Существует много других методов, меняющих параметры отбора в течение всей работы ГА. Элитизм применяется, если не используются другие методы, сохраняющие найденное хорошее решение.

Выбор способа кодирования обусловлен поставленной задачей и размером объекта поиска.

Операторы жизненного цикла (кроссинговера и мутации) определяются выбранным кодированием.

4. Модернизация ГА

Одной из серьезных проблем, возникающих при использовании генетических алгоритмов, является преждевременная сходимость. Не рекомендуется использовать классические ГА на маленьких популяциях, поскольку в популяциях с малым размером гены распространяются слишком быстро: все особи становятся похожими (популяция вырождается) еще до того, как найдено решение задачи. То есть новый генотип с лучшей оценкой быстро вытесняет менее хорошие комбинации генов, исключая тем самым возможность получения лучшего решения на их базе. Можно предложить три основных пути устранения преждевременной сходимости: увеличение размера популяции, применение самоадаптирующихся генетических операторов и создание «банка» заменяемых особей.

В первом случае, увеличивая размер популяции, можно надеяться на достижение многообразия генотипа в популяции. Но, с другой стороны,

увеличение числа особей ведет к увеличению занимаемой памяти и времени работы алгоритма. Данный подход может быть эффективен или при параллельных вычислениях, или при наличии достаточно простой целевой функции.

Второй, и самый распространенный способ, — использование самоадаптирующихся алгоритмов — является более эффективным. Самоадаптация заключается в применении динамических мутаций. Динамические мутации в зависимости от скрещивающихся особей меняют значение вероятности мутации, тем самым становится возможным самоуправление алгоритма. В таких случаях выбирается малый размер популяции.

В третьем подходе создается массив для сохранения особей, генотип которых был утерян при формировании новых поколений, и временами эти особи добавлятся в популяцию.

Ниже более подробно рассмотрим основные варианты динамических мутаций.

4.1. Самоадаптирующиеся алгоритмы

4.1.1. Неоднородная мутация

Одним из способов организации самоодаптирующегося алгоритма является применение неоднородной мутации [10]. Если мутирует ген y_i, то новое значение y_i' случайно генерируется на отрезке $[\min_i, \max_i]$:

$$y_i' = \begin{cases} y_i + (\max_i - y_i)\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right), & \text{если } q = 0, \\ \\ y_i - (y_i - \min_i)\left(1 - r^{\left(1-\frac{t}{T}\right)^b}\right), & \text{если } q = 1, \end{cases}$$

где q случайным образом принимает значения 0 или 1; r — случайное число, принимающее значение из диапазона $[0; 1]$; t — номер поколения; T — максимальное число поколений; b — некоторый параметр, обусловленный природой задачи; $\min_i$ и $\max_i$ — верхняя и нижняя границы для величины y_i.

4.1.2. Инцест

Стратегия инцеста используется как механизм самоадаптации оператора мутации. Она заключается в том, что «плотность мутации» (вероятность мутации каждого гена) определяется для каждого потомка на основании ге-

нетической близости его родителей. Например, это может быть отношение числа совпадающих генов родителей к общему числу генов хромосомы. В результате инцеста на начальных этапах алгоритма при высоком разнообразии генофонда популяции вероятность мутации будет предельно мала, т.е. практически будет происходить лишь скрещивание. При уменьшении разнообразия, возникающего в случае попадания алгоритма в локальный оптимум, вероятность мутации возрастет. Очевидно, что при полном схождении популяции алгоритм станет стохастическим, тем самым вероятность выхода популяции из локального оптимума возрастет.

Высокая вероятность мутаций гарантирует появление многообразия в популяции, но вполне возможно и разрушение хорошей особи при мутировании. Перьякс с соавторами предложили использовать помимо мутации-инцеста метод *супериндивидуального приближения* для отбора особей в новую популяцию [12–15]. При супериндивидуальном приближении выбирается наилучшая особь среди числа родителей и их потомков — элитная хромосома. Все хромосомы новой популяции являются копиями этой элитной хромосомы. Поэтому в популяции будут происходить сильные мутации.

Инцест также известен как мутация, зависящая от расстояния.

4.1.3. Критерий расстояния

В качестве расстояния в случае представления особи в виде бинарной строки (код Грея) берется расстояние Хемминга. Перьякс и другие [12–15] предложили другой критерий расстояния, основанный на критерии минимальной площади. И вместо использования абсолютного расстояния, они взяли относительное расстояние, которое в расчетах границ для каждой переменной индивидуально. Пусть $A = (a_1, \ldots, a_i, \ldots a_n)$ и $B = (b_1, \ldots, b_i, \ldots b_n)$ — два родителя, $\min_i$ и $\max_i$ — верхняя и нижняя границы для i-й переменной:

$$\operatorname{dist}(A, B) = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{a_i - b_i}{\max_i - \min_i} \right|^d,$$

где d — параметр, используемый для вычисления расстояния.

Пусть $M(A, B) = 1 - \operatorname{dist}(A, B)$ — вероятность мутации, связанная с расстоянием. Тогда $\operatorname{dist}(A, B) \in [0, 1]$ и $M(A, B) \in [0, 1]$. $M(A, B)$ достигает максимума при $\operatorname{dist}(A, B) = 0$; при идентичных родителях A и B получаем $M(A, B) = 1$. Кроссинговер двух идентичных индивидуумов при $M(A, B) = 1$, вызывает обязательную мутацию для каждой переменной-потомка. Очевидно, что такая мутация является очень большой, и поэто-

му авторы ввели множитель M_m, позволяющий скорректировать значение $M(A,B)$. Потом определяют эффективную вероятность мутации P_r: $P_r(A,B) = M(A,B)\,M_m = (1 - \mathrm{dist}(A,B))\,M_m$, где M_m — фиксированный параметр. В рассматриваемом классе задач оптимальными являются следующие значения параметров: $d = 0{,}2$ и $M_m = 0{,}9$.

4.1.4. Параметры вычислений

Для вычислений Перьякс и другие [12–15] использовали классические тестовые функции Де Ионга, широко применяемые для анализа эффективности ГА. Использовались следующие параметры:

- размер популяции $N = 20$;
- вероятность кроссинговера $P_{\mathrm{c}} = 0{,}8$;
- вероятность мутации воспроизводства $P_{\mathrm{m}}^{\mathrm{rep}} = 0{,}01$;
- максимальная вероятность мутации $M_{\mathrm{m}} = 0{,}9$.

Для каждой функции алгоритм запускался 100 раз.
Процесс работы генетического алгоритма представлен на рисунке 15.

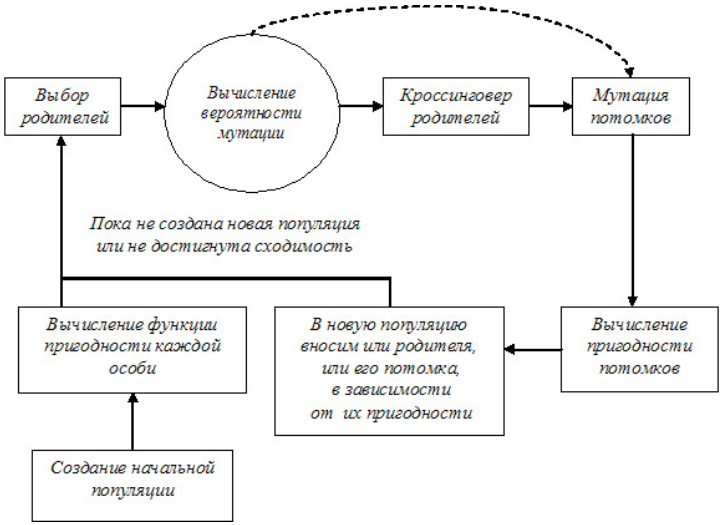

Рис. 15. ГА в работах Перьякса

5. Символьная модель ГА

5.1. Постановка задачи

Рассмотрим оптимизационную задачу (подробнее об оптимизационных задачах см. в *приложении А.2*): $\max f(x)$ с допустимым множеством

$$D = \{x = (x_1, x_2, \ldots, x_N) \,|x_i \in [a_i, b_i]\,, i = 1, 2, \ldots, N\},$$

где $f(x)$ — максимизируемая (целевая) скалярная многопараметрическая функция, которая может иметь несколько глобальных экстремумов, прямоугольная область D — область поиска, D — подмножество R^N. Предполагается, что о функции $f(x)$ известно лишь то, что она определена в любой точке области D. Никакая дополнительная информация о характере функции и ее свойствах (дифференцируемость, непрерывность и т.д.) в процессе поиска не учитывается.

Под решением поставленной задачи будем понимать вектор $x = (x_1, x_2, \ldots, x_N)$. Оптимальным решением будем считать вектор $x = x^*$, при котором целевая функция $f(x)$ принимает максимальное значение. Исходя из предположения о возможной многоэкстремальности $f(x)$, оптимальное решение может быть не единственным.

5.2. Символьная модель

Для того, чтобы построить пространство представлений под генетический алгоритм для задачи оптимизации необходимо дискретизировать пространство параметров скалярной функции $f(x)$. Параметры x обычно кодируются бинарной строкой s. Используя целевую функцию $f(x)$, можно построить функцию $\mu(s)$, отобразив, когда это необходимо, $f(x)$ на положительную полуось. Это делается для того, чтобы гарантировать прямое соотношение между значением целевой функции и приспособленностью решения. Впоследствии ГА работает именно с модифицированной целевой функцией $\mu(s)$. Таким образом, каждое возможное решение s, имеющее соответствующую приспособленность $\mu(s)$, представляет решение x. Обычно переход из пространства параметров D в хеммингово пространство бинарных строк осуществляется кодированием переменных $x_1, x_2, \ldots, x_N$ в двоичные целочисленные строки. Длина строк определяется требуемой точностью решения. При этом пространство параметров должно быть дискретизировано таким образом, чтобы расстояние между узлами дискретизации соответствовало требуемой точности. Предположим, по условию задачи с

функцией от двух переменных x_1 и x_2, определенной на прямоугольной области $D = \{0 < x_1 < 1;\ 0 < x_2 < 1\}$, требуется локализовать решение x^* с точностью по каждому из параметров 10^{-6}. Для достижения такой точности пространство параметров дискретизуется равномерной сеткой с $(b_i - a_i)/(10^{-6}) \sim 10^6$ узлами по каждой координате. Закодировать такое количество узлов можно $l = 20$ битами, где l определяется из условия $10^6 < 2^l + 1$. Получается, что общая длина бинарной строки кодировки для двумерной задачи составит $2 \times 20 = 40$ бит. При таком способе кодирования значения варьируемых параметров решений будут располагаться по узлам решетки, дискретизующей D. Соответственно, если кодировки двух решений будут совпадать, то будут совпадать и значения параметров обоих решений.

Во многих случаях такая модель может оказаться неэффективной. Кроме того, что она достаточно громоздка (каким будет хеммингово пространство поиска для задачи с сотней параметров?!). Практика показывает, что длинная кодировка повышает вероятность «преждевременной» сходимости. К тому же применение длинных кодировок вовсе не гарантирует, что найденное решение будет обладать требуемой точностью, поскольку этого, в принципе, не гарантирует сам ГА. Согласно этому, для того, чтобы применять ГА к задаче, сначала выбирается метод кодирования решений в виде строки. Фиксированная длина (l-бит) двоичной кодировки означает, что любая из 2^l возможных бинарных строк представляет возможное решение задачи. По существу, такая кодировка соответствует разбиению пространства параметров на гиперкубы, которым соответствуют уникальные комбинации битов в строке-хромосоме. Идея ГА состоит в том, чтобы, манипулируя имеющейся совокупностью бинарных представлений, с помощью ряда генетических операторов получать новые строки, т.е. перемещаться в новые гиперкубики. Получив бинарную комбинацию для нового решения, формируется вектор (операция декодирования) со значениями из соответствующего гиперкуба. Таким образом, каждое решение генетического алгоритма будет иметь следующую структуру (точка в пространстве параметров фенотип):

$$x = (x_1, x_2, \ldots, x_N) \in D \subset \mathbb{R}^N.$$

Бинарная строка s фиксированной длины, однозначно идентифицирующая гиперкуб разбиения пространства параметров (генотип) $s = (\beta_1, \beta_2, \ldots, \beta_l)$ принадлежит S, где S — пространство представлений бинарных строк длины l.

Скалярная величина μ, соответствующая значению целевой функции в точке x (пригодность): $\mu = f(x)$.

В терминологии, принятой в теории ГА, такую структуру принято называть особью.

Вообще могут существовать особи, обладающие различными фенотипическими признаками, но имеющие одинаковые генотипы (такое явление встречается в природе, например, у однояйцовых близнецов). Это позволяет использовать более крупное разбиение пространства параметров, сужая пространство бинарных строк S и делая при этом длину хромосомного набора короче. Многообразие точек, распределяемых в небольших гиперкубиках, позволяет достигать высокой точности.

5.3. Геометрическая интерпретация символьной модели

В предыдущем разделе было рассмотрено, каким образом будет осуществляться переход из евклидова пространства параметров в пространство представлений (бинарных строк). Рассмотрим эту процедуру на конкретном примере простой одномерной функции $f(x) = 10 + x \sin x$, определенной на отрезке [0, 10]. Пусть кодирование будет осуществляться бинарными строками длины 3 (см. рис. 16), то есть отрезок [0, 10] нужно разбить на $2^3 = 8$ подынтервалов, каждому из которых будет соответствовать уникальная двоичная комбинация, получаемая переводом номера подынтервала, считая слева направо, в двоичную систему. Длина каждого такого интервала будет $h = 10 : 8 = 1{,}25$.

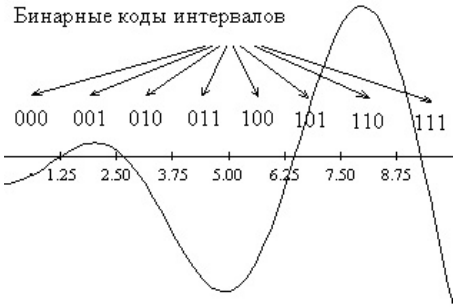

Рис. 16. Построение символьной модели для одномерной задачи с использованием трехбитового представления

Пространством поиска, таким образом, становится множество всех бинарных строк длины 3. Это пространство можно представить в виде трехмерного куба, вершинам которого соответствуют кодовые комбинации, расставленные так, что хэммингово расстояние между смежными вершинами равно 1 (см. рис. 17).

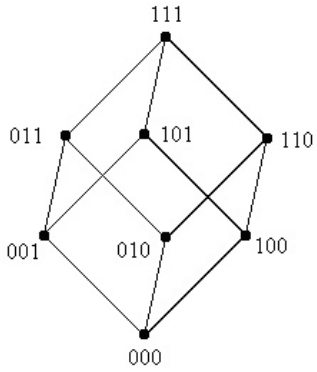

Рис. 17. Пространство поиска для трехбитового представления

Задача алгоритма поиска заключается в том, чтобы, следуя некоторому правилу, перемещаться в новые вершины этого куба, что будет соответствовать исследованию новых подинтервалов в пространстве D.

5.4. Шима

Хотя внешне кажется, что ГА обрабатывает строки, на самом деле при этом неявно происходит обработка шим, которые представляют шаблоны подобия между строками. ГА практически не может заниматься полным перебором всех представлений в пространстве поиска. Однако он может производить выборку значительного числа гиперплоскостей в областях поиска с высокой приспособленностью. Каждая такая гиперплоскость соответствует множеству похожих строк с высокой приспособленностью.

Шима (schema) — это строка длины l (что и длина любой строки популяции), состоящая из знаков алфавита $\{0; 1; *\}$, где $\{*\}$ — неопределенный символ. Каждая шима определяет множество всех бинарных строк длины l, имеющих в соответствующих позициях либо 0, либо 1, в зависимости

от того, какой бит находится в соответствующей позиции самой шимы. Шима, не содержащая ни одного неопределенного символа, является некоторой строкой. Шима с одним неопределенным символом описывает две бинарные строки, а с двумя — четыре строки. Например, шима, $10**1$, определяет собой множество из четырех пятибитовых строк {10001; 10011; 10101; 10111}. Нетрудно заметить, что шима с r-неопределенными символами описывает 2^r бинарных строк. С другой стороны, каждая стока длины m описывается 2^m шимами. Следовательно, в популяции из n таких строк число возможных шим может достигать $n2^m$! При этом большая часть шим вероятно будет менее приспособленной[6] остальных, что может привести к эпистазу. Поэтому рекомендуется содавать начальную популяцию из шим с высокой приспособленностью.

Все шимы различны между собой. Основными характеристиками шин являются порядок и длина.

Порядок шимы $o(S)$ (order) — это число фиксированных битов (0 или 1) в шиме S.

Определяющая длина $\delta(S)$ (defining length) — это расстояние между первым и последним фиксированными битами в шиме S. Длина шимы определяет концентрацию информации в шиме. Считается, что шима с одной фиксированной позицией имеет нулевую длину. Например, шина $S = (***001*110)$ имеет порядок $o(S) = 6$ и длину $\delta(S) = 10 - 4 = 6$.

Порядок и длина шим используются для определение вероятности мутации и кроссинговера соответственно.

В связи с тем, что более приспособленные особи (хромосомы) описываются шимой с большей приспособленностью, смысл работы ГА заключается в поиске двоичной строки определенного вида из всего множества бинарных строк длины m. Тогда пространство поиска составляет 2^m строк, а его размерность равна m. Шима соответствует некоторой гиперплоскости в этом пространстве. Данное утверждение можно проиллюстрировать следующим образом. Пусть разрядность хромосомы равна 3, тогда всего можно закодировать $2^3 = 8$ строк. Представим куб в трехмерном пространстве. Обозначим вершины этого куба трехразрядными бинарными строками так, чтобы метки соседних вершин отличались ровно на один разряд, причем вершина с меткой "000" находилась бы в начале координат (см. рис. 18).

Если взять шиму вида "$**0$", то она опишет левую грань куба, а шима "$*10$" — верхнее ребро этой грани. Очевидно, что шима "$***$" соответствует всему пространству. Если взять двоичные строки длиной 4 разряда, то

[6] Под приспособленностью шимы понимают среднюю приспособленность строк популяции, которые ей соответствуют.

разбиение пространства шимами можно изобразить на примере четырехмерного куба с поименованными вершинами (см. рис. 19).

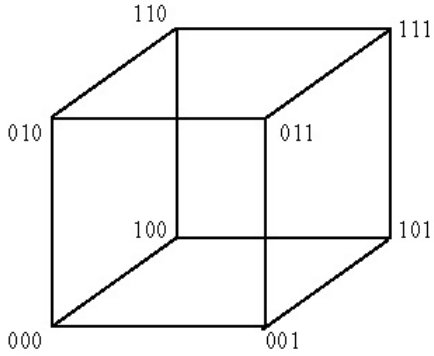

Рис. 18. Трехмерный куб

Здесь шиме "$*1**$" соответствует гиперплоскость, включающая задние грани внешнего и внутреннего куба, а шиме "$**10$" — гиперплоскость с верхними ребрами левых граней обоих кубов. Таким образом термины «гиперплоскость» и «шима» взаимозаменяемы.

Разбиение пространства поиска можно представить и по другому. Представим координатную плоскость, в которой по одной оси мы будем откладывать значения двоичных строк, а по другой — значение целевой функции (см. рис. 20).

Участки пространства, заштрихованные разным стилем, соответствуют разным шимам. Число K в правой части горизонтальной оси соответствует максимальному значению бинарной строки — "$111\ldots111$". Из рисунка видно, что шима "$0***\cdots*$" покрывает всю левую половину отрезка, шима "$**1*\cdots*$" — 4 участка шириной в одну восьмую часть, а шима "$0*10*\cdots*$" — левые половины участков, которые находятся на пересечении первых двух шим. Таким образом в этом случае происходит разбиение пространства.

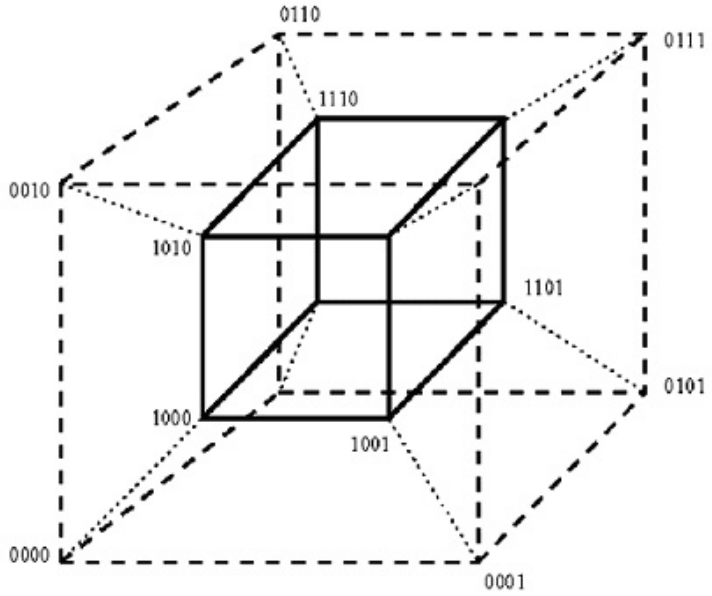

Рис. 19. Четырехмерный куб

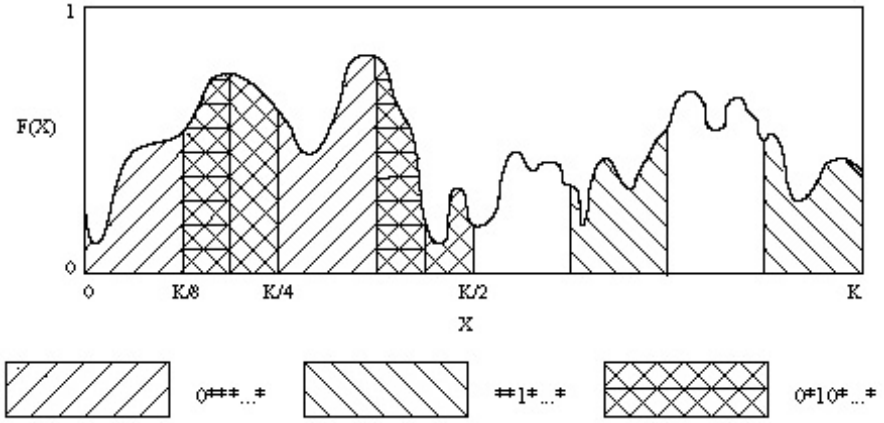

Рис. 20. Разбиение пространства

5.5. Строительные блоки

Строительные блоки — это шимы, обладающие:

- высокой пригодностью;
- низким порядком;
- короткой определенной длиной.

Пригодность шимы определяется как среднее пригодностей строк-особей, которые ее содержат. После процедуры отбора остаются только строки с более высокой пригодностью. Следовательно, строки, которые являются примерами шим с высокой пригодностью, выбираются чаще. Кроссинговер реже разрушает шимы с более короткой определенной длиной, а мутация реже разрушает шимы с низким порядком. Поэтому, такие шимы имеют больше шансов переходить из поколения в поколение. Холланд показал, что в то время, как ГА явным образом обрабатывает n строк на каждом поколении, неявно обрабатываются порядка n^3 таких коротких шим низкого порядка и с высокой приспособленностью (полезных шим – useful schemata (см. [8])). Он называл это явление неявным параллелизмом (implicit parallelism). Для решения реальных задач, присутствие неявного параллелизма означает, что большая популяция имеет больше возможностей локализовать решение экспоненциально быстрее популяции с меньшим числом особей.

5.6. Теорема шим

Теорема шим (The schema theorem) показывает, каким образом простой ГА экспоненциально увеличивает число примеров полезных шим или строительных блоков, что приводит к нахождению решения исходной задачи.

Пусть $m(H,t)$ — число примеров шимы H в t-ом поколении. Вычислим ожидаемое число примеров H в следующем поколении или $m(H,t+1)$ в терминах $m(H,t)$. Простой ГА каждой строке при отборе ставит в соответствие вероятность ее «выживания» пропорционально ее приспособленности (например, как в методе рулетки). Ожидается, что шима H может быть выбрана $m(H,t)(f(H)/f_{\text{ср}})$ раз, где $f_{\text{ср}}$ — средняя пригодность популяции, а $f(H)$ — средняя пригодность тех строк в популяции, которые являются примерами H.

Вероятность того, что одноточечный кроссинговер разрушит шиму равна вероятности того, что точка разрыва попадет между определенными битами. Вероятность же того, что H «переживает» кроссинговер не меньше

$1 - p_c(\delta(H)/l - 1)$, где p_c — вероятность кроссинговера. Эта вероятность — неравенство, поскольку шима сможет выжить, если в кроссинговере также участвовал пример подобной шимы.

Вероятность того, что H переживет точечную мутацию — $(1 - p_m)^{o(H)}$, где p_m — вероятность мутации. Это выражение можно аппроксимировать как $(1 - o(H))$ для малых p_m и $o(H)$. Произведение ожидаемого число отборов и вероятностей выживания известно как *теорема шим*:

$$\langle m(H, t+1)\rangle \geqslant m(H, t)\frac{f(H, t)}{\bar{f}(t)}\left[1 - p_c\frac{\delta(H)}{\ell - 1}\right](1 - p_m)^{o(H)}.$$

Теорема шим показывает, что строительные блоки растут по экспоненте, в то время шимы с приспособленностью ниже средней распадаются с той же скоростью. Голдберг в своих исследованиях теоремы шим выдвигает гипотезу строительных блоков, которая состоит в том, что «строительные блоки объединяются, чтобы сформировать лучшие строки» (см. [6]). То есть рекомбинация и экспоненциальный рост строительных блоков ведет к формированию лучших строительных блоков.

В то время как теорема шим предсказывает рост примеров хороших шим, сама теорема весьма упрощенно описывает поведение ГА. Прежде всего, $f(H)$ и f_{cp} не остаются постоянными от поколения к поколению. Во-вторых, теорема шим объясняет потери шим, но не появление новых. Новые шимы часто создаются кроссинговером и мутацией. Кроме того, в результате эволюции члены популяции становятся все более и более похожими друг на друга так, что разрушенные шимы будут сразу же восстановлены. Наконец, доказательство теоремы шим построено на элементах теории вероятности и, следовательно, не учитывает разброс значений. Во многих задачах разброс значений пригодности шимы может быть достаточно велик, делая процесс формирования шим очень сложным.

Существенная разница пригодности шимы может привести к сходимости к неоптимальному решению. Несмотря на простоту, теорема шим описывает несколько важных аспектов поведения ГА. Мутации с большей вероятностью разрушают шимы высокого порядка, в то время как кроссинговер с большей вероятность разрушает шимы с большей определенной длиной. Когда происходит отбор, популяция сходится пропорционально отношению приспособленности лучшей особи, к средней приспособленности в популяции: это отношение — *мера давления отбора* («selection pressure»). Увеличение или p_c, или p_m, или уменьшение давления отбора ведет к увеличенному осуществлению выборки или исследованию пространства поиска, но не позволяет использовать все хорошие шимы, которыми располагает

ГА. Уменьшение или p_c, или p_m, или увеличение давления выбора ведет к улучшению использования найденных шим, но тормозит исследование пространства в поисках новых хороших шим. Моделирование ГА предполагает сохранение равновесия ГА между тем и другим, что обычно известно как проблема «баланса исследования и использования».

Некоторые исследователи критикуют обычно быструю сходимость ГА, заявляя, что испытание огромных количеств перекрывающихся шим требует большей выборки и более медленной, более управляемой сходимости. Методология управления сходимость простого ГА до сих пор не выработана.

Недостатками теоремы шим является то, что она:

- примененяется только к каноническому ГА;
- не учитывает то обстоятельство, что кроссинговер и мутация могут не только разрушать шиму, но создавать ее из других шим. Поэтому в теореме шим присутствует знак неравенства;
- позволяет рассчитать долю шим в популяции только для следующего поколения, то есть при попытке подсчитать число строк, соответствующих данной шиме, через несколько поколений с использованием теоремы шим к успеху не приведет. Так получается, в частности, из-за пропорциональной стратегии отбора.

6. Преимущества и недостатки ГА

Генетические алгоритмы обладают рядом преимуществ, например, такими как:

- ГА не требуют никакой информации о поведении функции (например, дифференцируемости и непрерывности);
- разрывы, существующие на поверхности ответа, имеют незначительный эффект на полную эффективность оптимизации;
- ГА относительно стойки к попаданию в локальные оптимумы;
- ГА пригодны для решения крупномасштабных проблем оптимизации;
- ГА могут быть использованы для широкого класса задач;
- ГА просты в реализации;

- ГА могут быть использованы в задачах с изменяющейся средой.

В то же время существует ряд трудностей в практическом использовании ГА, а именно:

- с помощью ГА проблематично найти точный глобальный оптимум;
- ГА неэффективно применять в случае оптимизации функции, требующей большого времени на вычисление;
- ГА непросто смоделировать для нахождения всех решений задачи;
- не для всех задач удается найти оптимально кодирование параметров;
- в многоэкстремальных задачах ГА сталкивается с множеством аттракторов: на графике функции Растригина от одной переменной (рис. 21) видно, что истинный минимум достигается при $x = 0$;

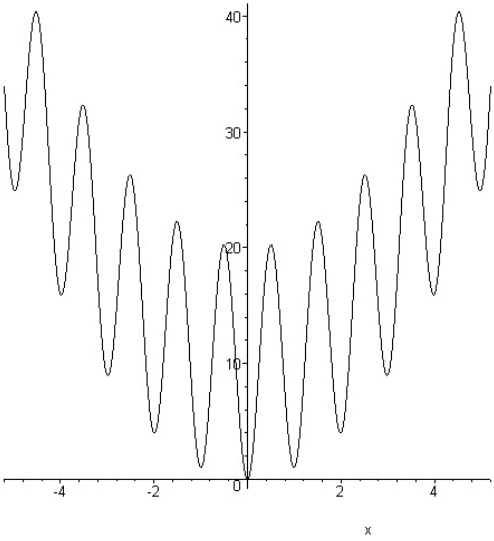

Рис. 21. Функция Растригина

- ГА трудно применить для изолированных функций. Изолированность («поиск иголки в стоге сена») — проблема для любого метода оптимизации, поскольку функция не предоставляет никакой информации, подсказывающей, в какой области искать максимум (рис. 22). Лишь

случайное попадание особи в глобальный экстремум может решить задачу (рис. 22);

Рис. 22. Изолированная функция

- дополнительный шум (noise) разбрасывает значения приспособленности шим, поэтому часто даже хорошие шимы малого порядка не проходят отбор, что замедляет поиск решения ГА (рис. 23).

Рис. 23. Зашумленная функция

- для некоторых функций шимы малого порядка уводят популяцию к локальному оптимуму. Такую характеристику функции называют обманчивостью (deception). Например, пусть строка состоит из 10 четырехбитных подстрок. Пусть u_i равно количеству единиц в i-ой подстроке. Зададим функцию $g(u)$ следующей таблицей:

u	0	1	2	3	4
$g(u)$	3	2	1	0	4

и пусть функция приспособленности равна сумме $g(u_i)$ по всем $i = 1 \ldots 10$:

$$f = \sum_{i=1}^{10} g(u_i).$$

Локальный максимум достигается при всех битах, равных 0, глобальный — при всех 1. В большинстве случаев при добавлении единицы

в подстроку приспособленность особи будет падать (за исключением случая, когда все остальные биты подстроки уже равны 1). При замене 1 на 0 она будет расти. Поэтому с большой вероятностью популяция сойдется к решению, при котором большинство подстрок будут состоять из всех нулей, и лишь некоторые из всех единиц. Однако это не будет глобальным максимумом. Из этого решения попасть в глобальный максимум, то есть заменить все нули единицами для ГА будет сложно. Рассмотренный пример схождения к ложному оптимуму демонстрирует явление эпистаза.

На данный момент ведутся поиски генетических стратегий, способных устранить эти недостатки. Устранение негативного влияния эпистаза и устранение ложного оптимума с применением разного рода мутаций, позволяющих выбить популяцию из тупика, было рассмотрено выше. В статьях А. Исаева говорится о возможности поиска всех решений. Он пишет: «... существует, по крайней мере, три класса задач, которые могут быть решены представленным алгоритмом:

- задача быстрой локализации одного оптимального значения,
- задача определения нескольких (или всех) глобальных экстремумов,
- задача описания ландшафта исследуемой функции, которая может сопровождаться выделением не только глобальных, но и локальных максимумов» (см. [17]).

Пространство поиска решения второй задачи достигается за счет некоторого сочетания параметров и достаточно большой численности популяции. При этом ГА сможет выделить несколько (или даже все) глобальные экстремумы. Для выделения нескольких глобальных максимумов, как пишут исследователи, больше всего подходит использование таких параметров, как, аутбридинг в сочетании с инбридингом, или элитный отбор или отбор с вытеснением (последний более надежный) для отбора особей в новую популяцию. Максимальная эффективность алгоритма достигается в сочетании аутбридинга в начале поиска для получения максимального широкого «исследования» и инбридинга в завершении поиска в целях уточнения решения в локальных группах.

7. Заключение

На основании проведенного анализа можно сделать выводы о том, что в настоящее время генетические алгоритмы являются мощным вычислительным средством в разнообразных оптимизационных задачах. ГА различаются с большинством обыкновенных оптимизационных и поисковых методов в 4 пунктах:

- ГА оперирует закодированным множеством параметров, а не с самими параметрами;
- ГА находит популяцию точек, а не отдельную точку;
- ГА использует значение объектной (целевой) функции, а не ее производную или другие вспомогательные значения;
- в ГА применяется вероятностное правило перехода, а не детерминистическое.

Для применения ГА необходимо прежде всего выбрать функцию пригодности (цели) адекватную задаче. Причем целевая функция должна иметь разнообразный рельеф, так как если на поверхности функции есть большие плоские участки, то ГА неэффективен. Это связано с тем, что многие особи в популяции при различии в генотипе не будут отличаться фенотипом, то есть несмотря на то, что особи различны, они имеют одинаковую пригодность, а значит алгоритм не имеет возможности выбрать лучшее решение и направление дальнейшего развития. Эта проблема известна как «проблема поля для гольфа», где все пространство абсолютно одинаково, за исключением лишь одной точки, которая и является оптимальным решением — в этом случае ГА просто остановится или будет блуждать абсолютно случайно. Другое требование к функции пригодности, состоит в требовании минимума вычислительных ресурсов при ее оценке, так как это влияет на скорость алгоритма.

В настоящее время ГА используются для решения таких задач, как:

- поиск глобального экстремума многопараметрической функции;
- аппроксимация функций;
- задачи о кратчайшем пути;
- задачи размещения;
- настройка искусственной нейронной сети;

- игровые стратегии;
- обучение машин.

A. Приложение

A.1. Теория Дарвина

Теория Дарвина биологической эволюции была изложена в 1859 г. в работе «Происхождение видов». Она рассматривает изменение популяции в определенной статичной или динамичной внешней среде. Под популяцией понимается большое количество особей одного вида, способных давать плодовитое потомство. Все члены популяции характеризуются индивидуальными внешними параметрами (фенотипом). Некоторые из параметров оказываются полезными для выживания и размножения, другие скорее вредят. Все внешние данные особи кодируются ее цепью ДНК (генотипом). Отдельные участки этой цепи (гены) определяют различные параметры особи.

В основе модели эволюции Дарвина лежат случайные изменения отдельных материальных элементов живого организма при переходе от поколения к поколению. Целесообразные изменения, которые облегчают выживание и производство потомков в данной конкретной внешней среде, сохраняются и передаются потомству (т.е. наследуются). Особи, не имеющие соответствующих характеристик, погибают, не оставив потомства или оставив его меньше, чем приспособленные (считается, что количество потомства пропорционально степени приспособленности).

Естественный отбор происходит в условиях конкуренции особей популяции, а иногда и различных видов, друг с другом за различные ресурсы, такие, например, как пища или вода. Кроме того, члены популяции одного вида часто конкурируют в привлечении брачного партнера. Те особи, которые наиболее приспособлены к окружающим условиям, проживут дольше и создадут более многочисленное потомство, чем их собратья. Это означает, что гены от высоко адаптированных или приспособленных особей будут распространяться в увеличивающемся количестве потомков на каждом последующем поколении. Некоторые потомки совместят в себе части цепи ДНК, отвечающие за наиболее удачные качества родителей, и, таким образом, окажутся еще более приспособленными. В итоге генотип слабо приспособленных особей с большой вероятностью исчезнет из генофонда популяции.

Изредка происходит мутация: некоторый случайный нуклеотид цепи

ДНК особи может измениться на другой. Если полученная цепь будет использоваться для создания потомства, то возможно появление у потомков совершенно новых качеств. Поэтому в результате естественного отбора возникает популяция из наиболее приспособленных особей, которая может стать основой нового вида.

Естественный отбор, скрещивание (кроссинговер) и мутация обеспечивают развитие популяции. Каждое новое поколение в среднем более приспособлено, чем предыдущее, т.е. оно лучше удовлетворяет требованиям внешней среды. Такой процесс называется эволюцией.

Генетические алгоритмы моделируют процесс эволюции в природе. Тем самым для решения задачи оптимизации многопараметрической функции и применяется теория Дарвина.

А.2. Некоторые понятия из теории оптимизации

Теория оптимизация занимается исследованием методов решения *оптимизационных задач*.

Оптимизационные задачи — это задачи, в которых требуется найти наилучшее решение; при этом, как правило, существуют различные ограничения на область изменения управляющих переменных, что не позволяет воспользоваться методами классического математического анализа, а требует применения разнообразных вычислительных методов.

Методы оптимизации — поиск экстремума функций (в практических задачах — критерий оптимальности) при наличии ограничений или без ограничений. Это прежде всего оптимальное проектирование (выбор наилучших номинальных технологических режимов, элементов конструкций, структуры технологических цепочек, условий экономической деятельности, повышение доходности и т.д.), оптимальное управление, построение нелинейных математических моделей объектов управления и т.д.

При рассмотрении любой оптимизационной задачи определяют:

1) *целевую (объектную) функцию*;

2) *управляющие параметры*.

Дадим определения этим понятиям.

Целевой функцией называется функция вида $f(x) \to \max$, $f(x) \to \min$ при наложенных условиях — ограничений.

Управляющие параметры — параметры, изменение которых влечет изменение значения целевой функции.

Если целевая функция зависит только от одного управляющего параметра, то такая оптимизационная задача называется *одномерной*, иначе — *многомерной*.

Ограничения в оптимизационной задаче могут быть:

- общими, если они накладываются на множество допустимых значений управляющих параметров, например, $x \geqslant 0$, где $x = (x_1, \ldots, x_n)$;
- специальными, если они накладываются на значения какой-либо функции, зависящей от значений управляющих параметров, например, $g_i(x) \leqslant 0$, где $x = (x_1, \ldots, x_n)$, $i = 1 \cdots$;
- типа неравенств (собственно ограничения), например, $f_i(x) \geqslant 0$, где $x = (x_1, \ldots, x_n)$, $i = 1 \cdots$ и f — целевая функция;
- типа равенств (связи), например, $f_i(x) = 0$, где $x = (x_1, \ldots, x_n)$, $i = 1 \cdots$ и f — целевая функция;
- активными, например, $x_{i\min} \leqslant x_i \leqslant x_{i\max}$. Активными называются такие ограничения, на границе которых находится решение (например, ограничения типа равенств);
- автономными.

Функции, задающие ограничения, могут формировать *допустимую область* (*допустимое множество*) с различными свойствами: монотонными, колебательными, с большей и малой кривизной и т.д.

Решение *допустимой* оптимизационной задачи состоит в выборе *оптимального решения* из *допустимого множества*.

Точка $x = (x_1, \ldots, x_n)$, удовлетворяющая ограничениям, называется *допустимым решением* оптимизационной задачи (в дальнейшем ОЗ).

Множество всех допустимых решений ОЗ называется *допустимым множеством* этой задачи.

Если ОЗ имеет хотя бы одно допустимое решение (т.е. ее допустимое множество не пусто), она называется *допустимой*, если ОЗ не имеет ни одного допустимого решения, то она называется *недопустимой*.

Точка x^0 называется *оптимальным решением* ОЗ, если:

- она является допустимым решением этой ОЗ;
- на этой точке целевая функция достигает глобального максимума (или минимума) среди всех точек, удовлетворяющих ограничениям.

При моделировании оптимизационной задачи можно пользоваться следующими условиями окончания поиска оптимального решения:

- малость изменения критерия оптимальности за 1 цикл;

- невозможность улучшения критерия оптимальности ни по одной переменной;

Для одномерной оптимизации: величина отрезка, содержащего оптимум, меньше заданной погрешности.

А.3. Кодирование Грея

Как известно в цифровой вычислительной технике основное место занимает двоичная система счисления (позиционная система с постоянным основанием, равным 2) и ее прямые производные (восьмеричная и шестнадцатеричная системы). Наряду с этими позиционными системами счисления используются и непозиционные. Самая распространенна из них — код Грея, известный также как рефлексный (отраженный) двоичный код. Этот код строится из двоичных цифр таким образом, что соседние в нем числа отличаются всегда только в одном разряде. Кодов с такой же характеристикой много, но для кода Грея имеется простой алгоритм перевода чисел в двоичный позиционный код и обратно.

А.3.1. Строение кода Грея

Код Грея — непозиционный код с одним набором символов (0 и 1) для каждого разряда. Таким образом, в отличие от римской системы счисления число в коде Грея не является суммой цифр. Чтобы показать соответствие последовательности чисел коду Грея можно воспользоваться таблицей, но есть и наглядное правило построения этой последовательности.

Младший разряд в последовательности чисел в коде Грея принимает значения 0 и 1, затем следующий старший разряд становится единичным и младший разряд принимает свои значения уже в обратном порядке (1, 0). Этим и объясняется название кода — отраженный. Соответственно, два младших разряда принимают значения 00, 01, 11, 10, а затем, при единичном следующем старшем разряде, те же значения в обратном порядке (10, 11, 01, 00). Ниже дана таблица, показывающая первые восемь чисел в двоичном коде и в коде Грея.

А.3.2. Алгоритмы преобразования кода Грея

Как сказано выше, алгоритм перевода чисел в коде Грея в позиционный код прост: каждый разряд в позиционном коде равен сумме по модулю 2

Таблица 8

Числа в коде Грея

число	*двоичный код*	*код Грея*
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

этого и всех более старших разрядов в коде Грея. Старшие разряды, соответственно, совпадают.

Перевод из позиционного кода в код Грея еще проще: каждый разряд в коде Грея равен сумме по модулю 2 этого и следующего старшего разряда в позиционном коде.

A.4. NP-полные (универсальные) задачи

Появление вычислительной техники привело к тому, что все реже приходится решать отдельную конкретную задачу, а все больше писать программы, рассчитанные на целый класс задач, получающихся одна из другой заменой ряда исходных данных. Поэтому имеет смысл говорить о сложности не для одной индивидуальной задачи I, а для массовой задачи, или проблемы P, соответствующей множеству индивидуальных задач.

Формально массовая задача P определяется общим списком всех параметров задачи (свободных параметров значения которых не заданы), формулировкой свойств, которым должен удовлетворять ответ (решение задачи). Индивидуальная задача I получается из P, если всем параметрам присвоить конкретные значения. Примером массовой задачи является задача коробейника: найти минимальный маршрут обхода группы городов с возвратом в начальную точку. В этой задаче входными параметрами будут число городов m и набор расстояний между ними, а требованием к решению — минимум суммарного расстояния, полученного путем обхода городов в каком-то порядке. Говорят, что алгоритм A решает массовую задачу P, если для любой индивидуальной задачи $I \in P$ алгоритм A применим к I (т.е. останавливается за конечное число шагов) и $I \in P$ алгоритм A дает решение задачи I.

Например, для P коробейника существует алгоритм, который решает ее на основе перебора всех маршрутов.

Большинство дискретных и комбинаторных задач допускает решение с помощью некоторого процесса перебора всех вариантов, однако число возможных вариантов растет экспоненциально в зависимости от размеров задачи (так, в задаче коробейника $m!$ маршрутов). Поэтому переборные алгоритмы решения массовых задач считаются неэффективными (могут решать лишь небольшие индивидуальные задачи). В отличие от них эффективными называются полиномиальные алгоритмы решения массовой задачи, то есть такие, которые решают произвольную $I \in P$ за время, ограниченное полиномом от «размера» I. Задачи P для которых существует полиномиальные алгоритмы решения называются полиномиальными. Другим примером массовых задач являются задачи распознавания свойств, предполагающие ответ «да» или «нет» в качестве решения. Такие задачи образуют широкий класс задач. Так, для любой задачи дискретной оптимизации можно указать аналогичную P распознавания свойств. В частности, если ввести в P задаче коробейника еще один параметр B — длину маршрута, то вопрос будет стоять следующим образом: существует ли маршрут длины, не превышающий B? Эта задача известна в математике под названием КМ (коммивояжера). Если для задачи P не существует полиномиального алгоритма, то она называется недетерминировано полиномиальной: $P \in NP$. В различных областях дискретной математики, комбинаторики, логики и т.п., известно множество задач, принадлежащих к классу NP-задач. Более того, если бы удалось отыскать эффективный алгоритм решения хотя бы одной из этих задач, то из этого следовало бы существование эффективных алгоритмов для всех остальных задач данного класса. На этом основано общепринятое мнение, что таких алгоритмов не существует.

A.5. Тестовые функции

Эффективность ГА зависит от выбора способов кроссинговера, мутации, выбора родительской пары, формирования новой популяции, а также от таких параметров как размер популяции, длина хромосомы. Оценить ГА можно с помощью разнообразных тестовых функций, например, функций Де Ионга. Все тестовые функции могут иметь различное число параметров (n). Поэтому имеет смысл запустить алгоритм для оптимизации некоторой функции сначала с небольшим n (например, 10 или 20), а затем с $n = 50, 100, 200, \ldots$ Это даст возможность проверить «масштабируемость» алгоритма. Так как генетические алгоритмы используют стохастичность, то для того, чтобы определить, насколько эффективен ГА, нужно запустить

его на одной и той же тестовой функции несколько раз, а потом взять средний результат. Ниже приводятся наиболее распространенные *тестовые функции* ($x = (x_1, x_2, \ldots, x_n)$) [**?**].

1. Сферическая функция, или функция Де Ионга 1 (*Sphere model, De Jong's function 1*):

$$f(x) = \sum_{i=1}^{n} x_i^2, \qquad x \in (-5{,}12; 5{,}12).$$

Глобальный минимум:

$$f(x) = 0; \qquad x_i = 0; \qquad i = 1 : n.$$

Локальных минимумов нет.

2. Гиперэллипсоидная функция с параллельными осями (*Axis parallel hyper-ellipsoid function*) похожа на функцию Де Ионга 1. По другому ее называют отягощенной сферической функцией. Следовательно, она выпукла и унимодальна:

$$f(x) = \sum_{i=1}^{n} i x_i^2, \qquad -5{,}12 \leqslant x_i \leqslant 5{,}12.$$

Глобальный минимум:

$$f(x) = 0; \qquad x_i = 0; \qquad i = 1 : n.$$

3. Повернутая гиперэллипсоидная функция (*Rotated hyper-ellipsoid function*). Данная функция получена поворотом гиперэллипсоидной функции, поэтому она выпукла и унимодальна.

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2, \qquad -65{,}536 \leqslant x_i \leqslant 65{,}536$$

Глобальный минимум:

$$f(x) = 0; \qquad x_i = 0; \qquad i = 1 : n$$

4. Ступенчатая функция Де Ионга (*Step function (De Jong)*):

$$f(x) = \sum_{i=1}^{n} |[x_i]|,\ x \in (-5{,}12; 5{,}12)\,,$$

где $|[x_i]|$ — модуль целой части.

5. Седло Розенброка, или функция Де Ионга 2 (*Rosenbrock's saddle (valley), De Jong's function 2*) представляет собой классическую оптимизационную функцию (она еще известна как «банановая функция»). Глобальный оптимум находится внутри параболической сильно вытянутой поверхности. Определение формы поверхности тривиально, однако сходимость к локальному минимуму становится трудной задачей и следовательно для данной функции приходится повторно проводить оценку с помощью других оптимизационных алгоритмов:

$$f(x) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \qquad x \in (-2{,}048; 2{,}048)\,.$$

Глобальный минимум:

$$f(x) = 0; \qquad x_i = 1, \qquad i = 1 : n.$$

6. Функция Де Ионга с гауссовым распределением случайных величин (*Gaussian quartic, De Jong*):

$$f(x) = \sum_{i=1}^{n} \left(x_i^4 + gauss(0,1)\right), \qquad x \in (-1{,}28; 1{,}28),$$

где $gauss(0,1)$ — функция, возвращающая случайную величину с нормальным распределением, с математическим ожиданием в 0 и дисперсией равной 1.

7. Функция Растригина (*Rastrigin's function*) является функцией одной переменной. Наличие косинуса влечет существование множества локальных минимумов:

$$f(x) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right), \qquad x \in (-5{,}12; 5{,}12)\,.$$

Глобальный минимум:

$$f(x) = 0; \qquad x_i = 0; \qquad i = 1 : n.$$

8. Функция Швефеля (*Schwefel's (Sine root) function*) имеет ложный глобальный минимум. Поэтому поисковые алгоритмы потенциально склоняются к сходимости в неверном направлении:

$$f(x) = 418{,}9829n + \sum_{i=1}^{n} \left(-x_i \sin \sqrt{|x_i|} \right), \qquad x \in (-500; 50).$$

Глобальный минимум:

$$f(x) = 0; \qquad x_i = 420{,}9829, \qquad i = 1 : n.$$

9. Функция Грайвенка (*Griewangk's function*) похожа на функцию Растригина. Она также имеет множество распространенных минимумов. Однако эти локальные минимумы регулярно распределены:

$$f\left(x\right) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right), x \in (-600; 600).$$

Глобальный минимум:

$$f(x) = 0; \qquad x_i = 0; \qquad i = 1 : n.$$

10. Функция Эккли (*Ackley's function*):

$$f(x) = 20 + e - 20 \exp\left(-0{,}2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos\left(2\pi x_i\right) \right),$$

$$x \in (-30; 30).$$

Глобальный минимум:

$$f(x) = 0; \qquad x_i = 0; \qquad i = 1 : n.$$

11. Сумма различных степеней (*Sum of different power*) — часто используемая тестовая функция:

$$f(x) = \sum_{i=1}^{n} |x_i|^{i+1}, \qquad -1 \leqslant x_i \leqslant 1.$$

Глобальный минимум:

$$f(x) = 0; \qquad x_i = 0; \qquad i = 1 : n.$$

12. Функция Михалевича (*Michalewicz's function*) — это мультимодальная тестовая функция с $n!$ локальными экстремумами. Параметр m определяет «крутизну» (steepness) оврагов или хребтов. Большое m делает поиск труднее. Для очень больших m поведение функции функции в данной точке дает ничтожную информацию о локальных и глобальных экстремумах), так как рельеф функции представляет собой обширные плоские участки (овраги и плато):

$$f(x) = -\sum_{i=1}^{n} \left(\sin(x_i) \cdot \sin^{20}\left(\frac{i}{\pi} x_i^2 \right) \right), 0 \leqslant x_i \leqslant \pi.$$

Глобальный минимум:

$$f(x) = -4{,}687 \qquad (n = 5);$$
$$f(x) = -9{,}66 \qquad (n = 10).$$

13. Функция Бранинса (*Branins's rcos function*). Данная функция двух переменных имеет глобальный минимум в трех точках:

$$f(x_1, x_2) = \left(x_2 - \frac{5{,}1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10,$$
$$-5 \leqslant x_1 \leqslant 10, \qquad 0 \leqslant x_2 \leqslant 15.$$

Глобальный минимум:

$$f(x_1, x_1) = 0{,}397887,$$
$$(x_1, x_1) = \{(-\pi, 12{,}275); (\pi, 2{,}275); (9{,}42478; 2{,}475)\}.$$

14. Функция Изома (*Easom's function*). У данной унимодальной тестовой функции глобальный минимум находится в маленькой области относительно всего поискового пространства. Функция была инвертирована для минимизации:

$$f(x_1, x_2) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2 + (x_2-\pi)^2\right),$$
$$-100 \leqslant x_i \leqslant 100, \qquad i = 1:2.$$

Глобальный минимум:

$$f(x_1, x_2) = -1; \qquad (x_1, x_2) = (-\pi, \pi).$$

15. Функция Голдстейна (*Goldstein–Price function*):

$$f(x_1, x_2) = [1+(1+x_1+x_2)^2 \cdot (19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)] \times$$
$$\times [30 + (2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)];$$
$$-2 \leqslant x_i \leqslant 2, \qquad i = 1:2.$$

Глобальный минимум:

$$f(x_1, x_2) = 3; \qquad (x_1, x_2) = (0, -1).$$

16. Шестигорбая функция (*Six–hump camel back function*) имеет глобальный минимум в двух точках. По границе области содержащей глобальные минимумы находится шесть локальных минимумов:

$$f(x_1, x_2) = \left(4 - 2{,}1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right);$$
$$-3 \leqslant x_1 \leqslant 3, \qquad -2 \leqslant x_2 \leqslant 2.$$

Глобальный минимум:

$$f(x_1, x_2) = -1{,}0316;$$
$$(x_1, x_2) = \{(-0{,}0898; 0{,}7126); (0{,}0898, -0{,}7126)\}.$$

B. Генетические алгоритмы в MATLAB 7.0.1

B.1. Общие сведения

Одним из новшеств MATLAB 7.0.1 является тулбокс *Genetic Algorithm and Direct Search Toolbox*, который предназначен для расширения функциональных возможностей пакета, в частности *Optimization Toolbox*, генетическими алгоритмами. Такие алгоритмы чаще всего используются в случае, когда искомая целевая функция является разрывной, существенно нелинейной, стохастической и не имеет производных или эти производные являются недостаточно определенными. Работать с генетическими алгоритмами теперь можно в двух тулбоксах.

Собственно генетические алгоритмы относятся к разделу *Genetic Algorithm* и вызываются из командной сроки с помощью *gatool* или *ga*.

Генетические алгоритмы и их комбинации с другими оптимизационными методами можно найти в разделе *Direct Search Toolbox*. Для этого в командной строке необходимо набрать `psearchtool` (от poll search tool — средство поиска упорядоченным опросом).

Рассмотрим первый вариант работы с ГА. Существуют 4 основные функции для работы с алгоритмом:

ga — функция для нахождения минимума целевой функции;

gaoptimget — возвращает параметры используемого генетического алгоритма;

gaoptimset — устанавливает параметры генетического алгоритма;

gatool — открывает окно *Genetic Algorithm Tool*.

Для того, чтобы применить ГА к поставленной функции цели, необходимо в первую очередь записать её в M-file и сохранить в текущей папке.

B.2. Функция ga

Функция ga вызывается в командной строке согласно нижеприведенному синтаксису:

```
[x fval] = ga(@fitnessfun, nvars, options)
```

здесь

`fitnessfun` — имя M-file, содержащего поставленную целевую функцию;

`nvars` — число независимых переменных в целевой функции;

`options` — структура, содержащая параметры используемого ГА. Если параметры не изменять, то их значения возьмутся по умолчанию;

Результаты вычислений сохранятся в переменных:
`fval` — окончательное значение целевой функции;
`x` — точка, в которой достигнуто оптимальное значение.
Существуют и другие варианты вызова функции ga:

```
x = ga(fitnessfun, nvars)

x = ga(fitnessfun, nvars, options)

x = ga(problem)

[x, fval] = ga(...)

[x, fval, reason] = ga(...)

[x, fval, reason, output] = ga(...)

[x, fval, reason, output, population] = ga(...)

[x, fval, reason, output, population, scores] = ga(...)
```

Описание `x = ga(fitnessfun, nvars)` применяется для решения оптимизационной задачи, `fitnessfun` — минимизируемая целевая функция и `nvars` — длина вектора решений x, соответствующего наилучшей особи.

`x = ga(fitnessfun, nvars, options)` применяется для решения оптимизационной задачи, используя параметры (`options`) алгоритма.

`x = ga(problem)` находит минимум задачи, структура которой описывается тремя полями:

`fitnessfcn` — целевая функция;

`nvars` — число независимых переменных целевой функции;

`options` — параметры структуры ГА, задаваемые функцией *gaoptimset*.

`[x, fval] = ga(...)` возвращает `fval`, значение целевой функции по `x`.

`[x, fval, reason] = ga(...)` возвращает `reason` — строку, содержащую параметры остановки алгоритма.

`[x, fval, reason, output] = ga(...)` возвращает `output` — совокупность сведений о каждом поколении и другой информации о реализации алгоритма. Структура output состоит из следующих полей:

`Randstate` или (`randnstate`) — начальное состояние популяции сгенерированное случайными числами. (Различие функций состоит в разных выводах.);

`generations` — количество вычисляемых поколений;

`funccount` — количество вычислений функции;

`message` — параметры остановки алгоритма. Это сообщение выводит несколько аргументов завершения алгоритма.

`[x, fval, reason, output, population] = ga(...)` возвращает матрицу популяции, строки которой соответствуют особи конечной популяции.

`[x, fval, reason, output, population, scores] = ga(...)` возвращает расчеты финальной популяции.

Замечание: для всех оптимизационных задач популяция должна быть представлена в виде вещественных чисел. Функция ga не работает для функций с комплексными переменными. Для решения задач включающих оптимизационные числа нужно записать целевую функцию в виде допустимого вещественного вектора, отделив вещественные и комплексные части.

Пример

```
[x fval, reason] = ga(@rastriginsFcn, 10)

x =

Columns 1 through 7

0.9977 0.9598 0.0085 0.0097 -0.0274 -0.0173 0.9650

Columns 8 through 10

-0.0021 -0.0210 0.0065

fval =

3.7456

reason =

generations
```

B.3. Функция gaoptimset

Для настройки генетического алгоритма используется функция *gaoptimset*. Она позволяет построить ГА комбинируя, операторы по желанию пользователя. Синтаксис данной функции выглядит следующим образом.

Синтаксис

```
options = gaoptimset

gaoptimset

options = gaoptimset('param1',value1,'param2',value2,...)

options = gaoptimset(oldopts,'param1',value1,...)

options = gaoptimset(oldopts,newopts)
```

Описание

`options = gaoptimset` (здесь аргументы не вводятся) с помощью данной конструкции задается структура ГА, отличная от структуры ГА по умолчанию.

`gaoptimset` не требует ввода или вывода аргументов. В результате сформирует список параметров и их действительных значений.

`options = gaoptimset('param1',value1,'param2',value2,...)` генерирует структуру с множеством параметров их значений. Для нескольких неспециальных параметров можно использовать значения по умолчанию.

`options = gaoptimset(oldopts,'param1',value1,...)` создает копию `oldopts`, модифицированную выбранными специальными параметрами и их значениями.

`options = gaoptimset(oldopts,newopts)` комбинирует параметры существующей структуры `oldopts`, с параметрами новых структур `newopts`. Некоторые параметры с ненулевыми значениями в `newopts` могут заменить или быть присвоены старым параметрам в `oldopts`.

Опции

В следующей таблице приведен список параметров для функции `gaoptimset`. В фигурных скобках {} берутся значения по умолчанию.

Таблица 9

Список параметров функции `gaoptimset`

Опции	**Описание**	**Значения**
CreationFcn	Используется для создания начальной популяции	{@gacreationuniform}
CrossoverFraction	Вероятность кроссинговера, не включает элитных потомков, полученных при кроссинговере	Положительные числа | {0.8}
CrossoverFcn	Вид кроссинговера	@crossoverheuristic (случайный кроссинговер) {@crossoverscattered} @crossoverintermediate @crossoversinglepoint (одноточечный кроссинговер) @crossovertwopoint (двуточечный кроссинговер)
EliteCount	Положительное целое число, определяющее количество особей текущей популяции, которые будут скопированы в новое поколение	Положительное целое | {2}
FitnessLimit	Число. Если функция пригодности достигнет значения FitnessLimit, то алгоритм остановится	Число | {-Inf}
FitnessScalingFcn	Устанавливается, если значениями функции пригодности являются числа	@fitscalinggoldberg {@fitscalingrank} @fitscalingprop @fitscalingtop
Generations	Положительное число, определяющее максимальное число итераций алгоритма	Положительное целое |{100}

Опции	**Описание**	**Значения**
PopInitRange	Матрица или вектор, определяющие особи в начальной популяции	Матрица или вектор | [0;1]
PopulationType	Строковое выражение, описывающее тип данных в популяции	'bitstring' | 'custom' | {'doubleVector'}
HybridFcn	Устанавливает дальнейшую оптимизацию по завершению работы ga	Установочная функция | {[]}
InitialPopulation	Начальная популяция	Положительные числа | {[]}
InitialScores	Начальные точки	Вектор-столбец | {[]}
MigrationDirection	Направление миграции	'both' | {'forward'}
MigrationFraction	Число между 0 и 1, соответствующее доли мигрирующих особей для каждой подпопуляции	Число | {0.2}
MigrationInterval	Положительное целое соответствующее количеству поколений между миграциями	Положительное целое | {20}
MutationFcn	Устанавливается в случае мутации потомков	@mutationuniform {@mutationgaussian}
OutputFcns	Массив, содержащий функции, которые ГА вызывал каждую итерацию	массив | {[]}
OutputInterval	Положительное целое число поколений между последовательными вызовами функции вывода	Положительное целое | {1}

Опции	Описание	Значения
PlotFcns	Массив для сохранения результатов, полученных в ходе вычислений алгоритма. В дальнейшем используется для построения графика	@gaplotbestf @gaplotbestgenome @gaplotdistance @gaplotexpectation @gaplotgeneology @gaplotselection @gaplotrange @gaplotscorediversity @gaplotscores @gaplotstopping | {[]}
PlotInterval	Положительное целое число поколений между последовательными вызовами функции plot	Положительное целое | {1}
PopulationSize	Размер популяции	Положительное целое | {20}
SelectionFcn	Устанавливает функцию отбора родителей для кроссинговера и мутации потомков	@selectiongoldberg @selectionrandom {@selectionstochunif} @selectionroulette @selectiontournament
StallGenLimit	Положительное целое. Алгоритм остановится, если за StallGenLimit последовательных итераций не произойдет улучшение функции цели	Положительное целое | {50}
StallTimeLimit	Положительное целое. Алгоритм остановится, если за StallTimeLimit секунд не произойдет улучшение функции цели	Положительное целое | {20}
TimeLimit	Положительное число. Алгоритм остановится после TimeLimit секунд с момента начала работы	Положительное число | {30}
Vectorized	Строка, определяющая векторизацию функции цели	'on' | {'off'}

Например:

1. Функция `gaoptimget`

   Функция `gaoptimget` обладает следующим синтаксисом:

```
val = gaoptimget(options, 'name');
```

2. Функция `gatool`.

3. Другой наиболее наглядный способ работы с ГА заключается в вызове окна с помощью функции `gatool`.

   Окно тулбокса представлено на рис. 24.

   Как видно из рисунка, окно сопровождено справкой по всем компонентам, и работа пользователя заключается в виде установки параметров и нажатии кнопки «start». Результат будет тем же, что и в случае последовательного применения функций `gaoptimset` и `ga`. В разделе `plots` можно выбрать переменные, изменение которых будет отображаться графически.

B.4. Векторизация целевой функции

В тулбоксе ГА предусмотрена векторизация функций, благодаря чему вычисления происходят заметно быстрее. Смысл данного метода заключается в том, что в качестве параметров функции выступают вектора, тогда для текущей популяции целевая функция будет вызываться лишь один раз, вычисляя пригодности всех особей. Например, рассмотрим функцию

$$f(x_1, x_2) = x_1^2 - 2x_1x_2 + 6x_1 + x_2^2 - 6x_2.$$

Запишем для неё M-file, используя следующий код:

```
z =x(:,1).\ 2 - 2*x(:,1).*x(:,2) + 6*x(:,1) +x(:,2).\ 2
-6*x(:,2);
```

Здесь `x(:, 1)` представляет собой вектор, а `.\` и `.*` — операции поэлементного соответственно возведения в степень и умножения.

В окне тулбокса в списке `Vectorize option` следует установить значение `On`.

Убедиться в эффективности векторизации можно на примере функции Растригина. В командной строке введем следующее выражение:

```
tic;ga(@rastriginsfcn,20);toc
```

В результате можно узнать время, затраченное на вычисления:

```
elapsed_time =
4.3660
```

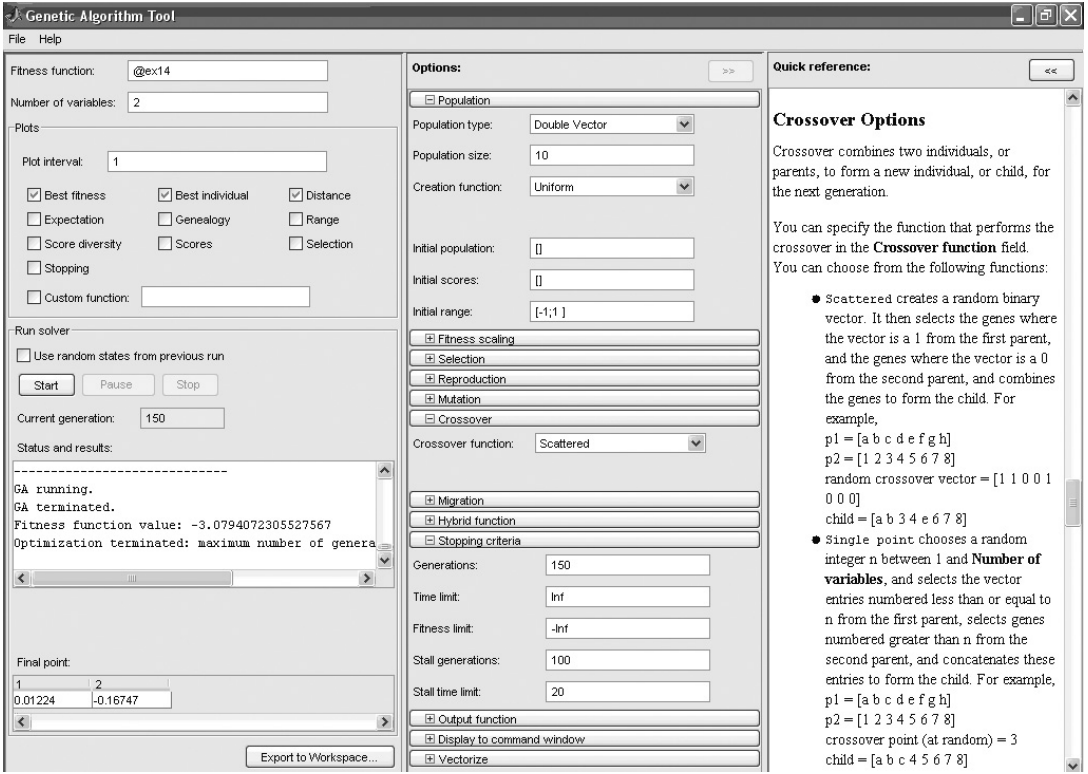

Рис. 24. Окно тулбокса по генетическим алгоритмам

Проделаем тоже самое, но используя векторизацию функции Растригина:

```
options=gaoptimset('Vectorize','on');
tic;ga(@rastriginsfcn,20,options);toc \
```

Узнаем время затраченное на векторизацию:

```
elapsed_time = 0.581
```

Как видно метод векторизации работает на много быстрее!

B.5. Лабораторная работа

Перед выполнением лабораторных работ внимательно прочтите решение лабораторной работы № 1!

B.5.1. Лабораторная работа № 1

Задание № 1

Минимизировать функцию одной переменной

$$f(x) = 8x - 16 - 12\sqrt[3]{(x+4)^2}.$$

Решение

Напишем M-file для данной функции и сохраним его в текущей папке под именем ex2.m.

```
function y = ex2(x)
y=-12*(x+4)\ (2/3)+8*x-16;
```

Вызовем окно тулбокса с помощью *gatool*.

В поле *fitness function* введем имя целевой функции @ex2

Установим значения параметров ГА: количество особей в популяции = 10, количество поколений = 100 (в окне критерия остановки алгоритма), начальный отрезок = $[-4; 1]$. В разделе *plots* установим флажки для *best fitness*, *best individual*, *distance*. Щелкнем по кнопке *start*.

В результате завершения процесса в окне *final point* появится значение переменной **x**, соответствующее минимуму функции, а в окне *status and result* можно увидеть найденное минимальное значение целевой функции.

Для данной задачи результаты получились следующие минимум функции достигается в точке $x = -2.9972$ и $f(-2.9972) = -51.99998959105959$ (рис. 25).

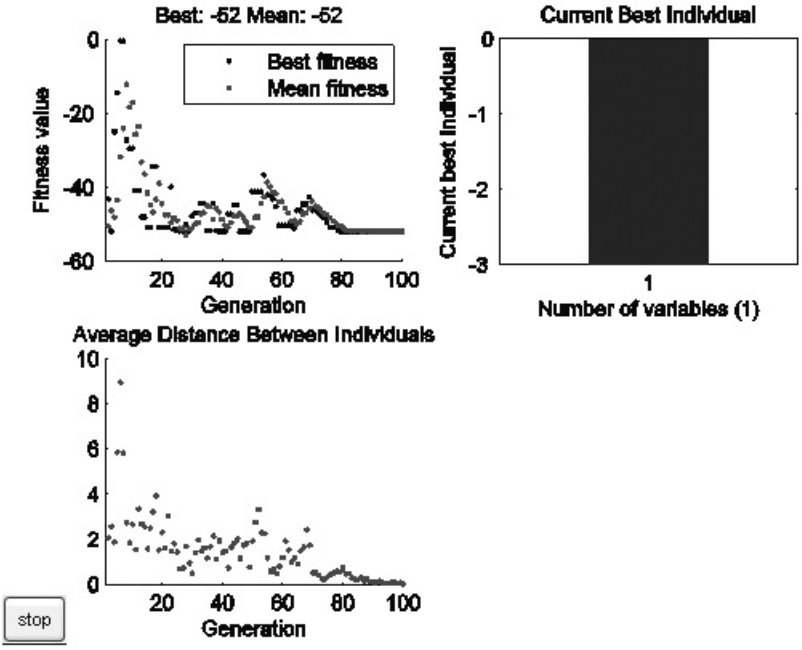

Рис. 25. Графический анализ решения

Первый рисунок отображает изменение значение целевой функции. Видно, что, начиная с 80 популяции, алгоритм сошелся к решению. На втором рисунке изображена наилучшая особь. Третий рисунок соответствует изменению расстояния между особями в поколениях. Особи становятся одинаковыми (хеммингово расстояние = 0) в последних 18 поколениях. ГА нужно запустить несколько раз, а потом выбрать оптимальное решение. Это связано с тем, что начальная популяция формируется с использованием генератора случайных чисел.

Убедиться в правильности решения можно, построив график функции (рис. 26).

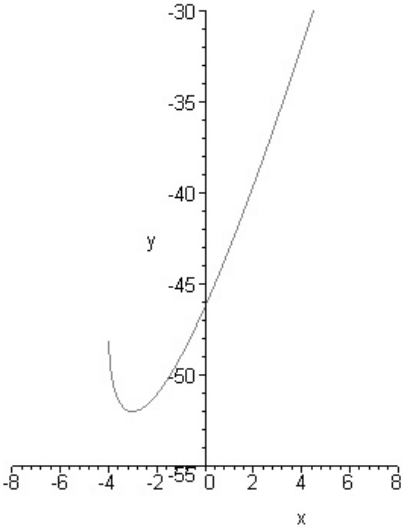

Рис. 26. График функции

То же самое можно было бы получить, используя функции *gaoptimset* и *ga*. Чтобы посмотреть *M*-File выберете в меню «File» окна «Genetic Algoritm Tool» команду «Generate *M*-file», сохраните файл под другим именем и просмотрите код. Для данной задачи получили:

```
function [X,FVAL,REASON,OUTPUT,POPULATION,SCORES] = ex2q

% This is an auto generated M file to do optimization
% with the Genetic Algorithm and
```

```
% Direct Search Toolbox. Use GAOPTIMSET for default
% GA options structure.

% Fitness function

fitnessFunction = @ex2;

% Number of Variables

nvars = 1 ;

% Start with default options

options = gaoptimset;

% Modify some parameters

options = gaoptimset(options,'PopInitRange',[-4 ; -1 ]);

options = gaoptimset(options,'PopulationSize',10);

options = gaoptimset(options,'MutationFcn',
{@mutationgaussian 1 1});

options = gaoptimset(options,'Display','off');

options = gaoptimset(options,'PlotFcns', {@gaplotbestf
@gaplotbestindiv @gaplotdistance });

% Run GA

[X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =
ga(fitnessFunction,nvars,options);
```

Задание № 2

Максимизировать функцию двух переменных:

$$z(x, y) = \exp(-x^2 - y^2) + \sin(x + y).$$

Решение

Тулбокс по ГА решает только задачи минимизации, для нахождения максимума функции $f(x)$ следует минимизировать функцию $-f(x)$. Это объясняется тем, что точка минимума $-f(x)$ является некоторой точкой $f(x)$, в которой достигается максимум.

Напишем M-file для функции $z(x) = -f(x)$ и сохраним его в текущей папке под именем `ex13.m`:

```
function z = ex13(x)

z=-(exp(-x(1)\ 2-x(2)\ 2)+sin(x(1)+x(2)));
```

Вызовем окно тулбокса с помощью *gatool*.

В поле *fitness function* введем имя целевой функции `@ex13`.

Установим значения параметров ГА: количество переменных = 2, количество особей в популяции = 10, количество поколений = 100 (в окне критерия остановки алгоритма), начальный отрезок = [-1; 3]. Для построения графиков в разделе *plots* установим флажки для *best fitness*, *best individual*, *distance*. Щелкнем по кнопке *start*.

В результате завершения процесса в окне *final point* появится значение переменной x, соответствующее минимуму функции, а в окне *status and result* можно увидеть найденное минимальное значение целевой функции $z(x)$.

Для данной задачи результаты получились следующие: максимум функции достигается в точке $x = 0.46419$, $y = 0.42406$ и $f(0.46419; 0.42406) = 1.449$.

Поверхность функции представлена на рис. 27.

В.5.2. Лабораторная работа № 2

Задание № 1

Найти максимум функции $y(x) = 3\sqrt[3]{(x+4)^2} - 2x - 8$.

Задание № 2

Найти минимум функции $z(x, y) = (y - 3)\exp(-x^2 - y^2)$.

Задание № 3

Построить тестовую функцию Эккли для двух переменных.

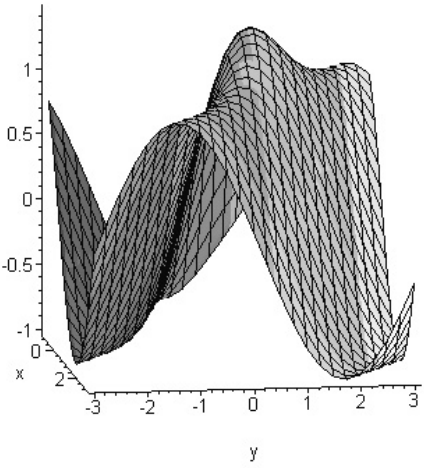

Рис. 27. Поверхность функции. ЛР1

В.5.3. Лабораторная работа № 3

Задание № 1
Найти максимум функции $y(x) = \frac{6\sqrt[3]{6(x-3)^2}}{(x-1)^2+8}$.
Задание № 2
Найти минимум и максимум функции $z(x, y) = x\exp(-x^2 - y^2)$.
Задание № 3
Построить тестовую функцию Михалевича для двух переменных.

В.6. Ответы

Лабораторная работа № 2
Задание № 1
$x = -3.99927, y(x) = 0.999$ (рис. 28).
M-File:

```
function y = ex1(x)
y=-(3*(x+4)\ (2/3)-2*x-8);
```

График изображен на рисунке 29.

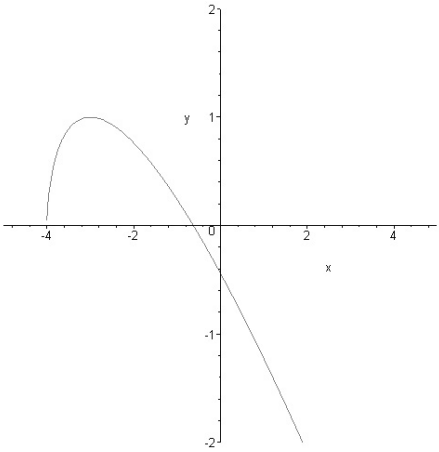

Рис. 28. График функции. ЛР2

Задание № 2
$x = 0.01224, y = -0.16747, z(x, y) = -3.0794$ (рис. 29).
M-File:

```
function z = ex14(x)

z=(x(2)-3)*exp(-x(1)\ 2-x(2)\ 2);
```

Для сходимости алгоритма выберите 150 поколений.

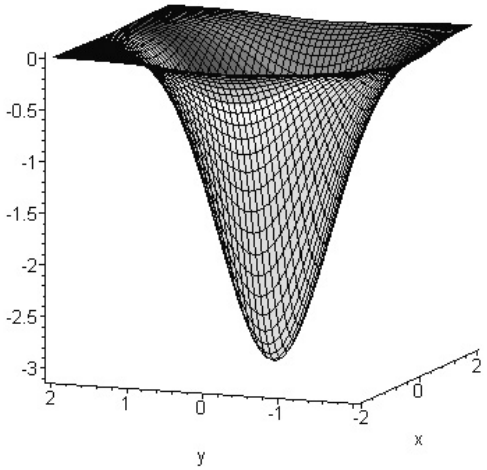

Рис. 29. Поверхность функции. ЛР2

Задание № 3
Рисунок 30.
Лабораторная работа № 3
Задание № 1
$x = 4.97881, y(x) = 1.31$ (рис. 31).
M-File:

```
function y = ex3(x)

y=-6*(6*(x-3))\ (2/3)/(8+(x-1)\ 2);
```

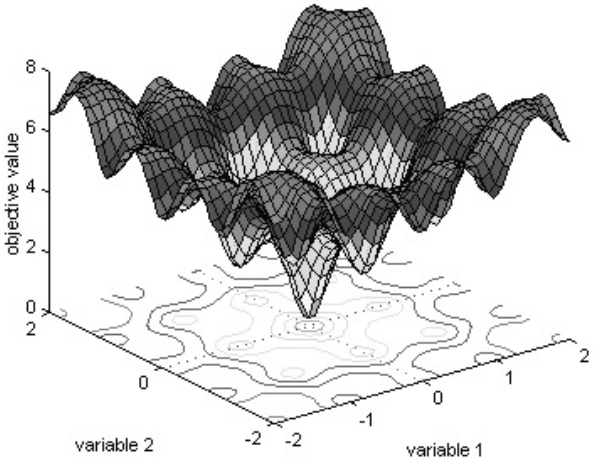

Рис. 30. Поверхность функции Эккли

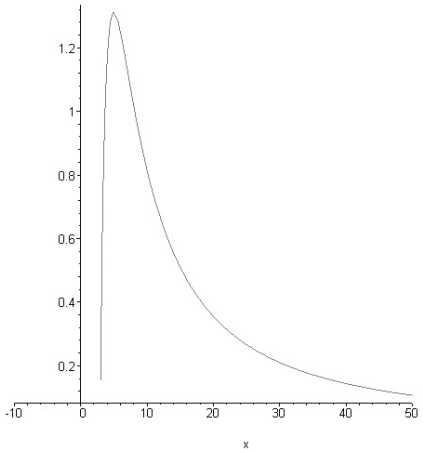

Рис. 31. График функции. ЛР3

Задание № 2

Минимум $x = -0.7428, y = -0.01912, z(x, y) = -0.4276$.

M-File:

```
function z = ex12(x)

z=x(1)*exp(-x(1)\ 2-x(2)\ 2);
```

Максимум $x = 0.7442, y = -0.07576, z(x, y) = 0.425275$ (рис. 32).

M-File:

```
function z = ex122(x)

z=-x(1)*exp(-x(1)\ 2-x(2)\ 2);
```

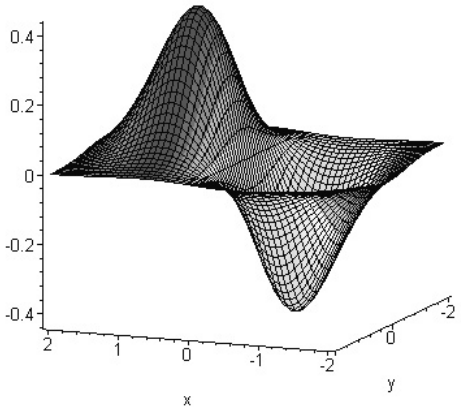

Рис. 32. Поверхность. ЛР3

Задание № 3

Поверхность — на рисунке 33.

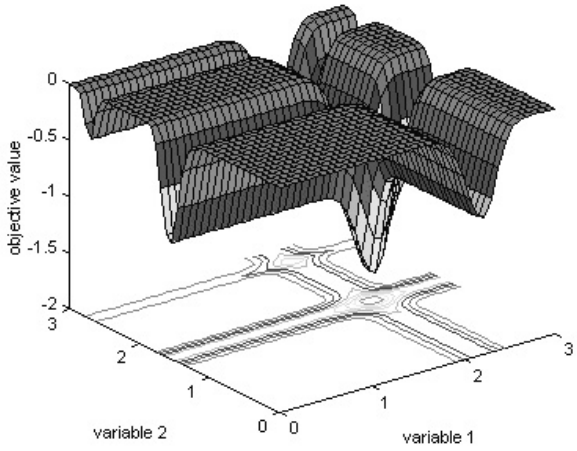

Рис. 33. Поверхность функции Михалевича

Также вы можете сконструировать другие ГА, моделируя операторы (выбор родительских пар, кроссинговер, мутация, миграция, отбор особей в новую популяцию, критерии завершения алгоритма) и параметры алгоритма. Дальнейшее изучение оптимизации с помощью ГА можно продолжить, рассматривая *Direct Search Toolbox*. Для вызова окна можно в командной строке просто прописать *psearchtool*.

Список литературы

1. Батищев, Д.И. Генетические алгоритмы решения экстремальных задач [Текст]/ Д.И. Батищев ; Нижегородский госуниверситет. — Нижний Новгород : 1995.с. — 62с.

2. Дарвин Ч. О происхождении видов путём естественного отбора или сохранении благоприятствуемых пород в борьбе за жизнь [Текст]/ Ч. Дарвин. — М.: АН СССР, 1939. — Т.3.

3. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы [Текст]/ Под ред. В.М. Курейчика. — 2-е изд., испр. и доп. — М.: ФИЗМАТЛИТ, 2006. — 320 с. — ISBN 5-9221-0510-8.

4. Darrel Whitley: A Genetic Algorithm Tutorial; November 10, 1993; Technical Report CS-93-103 (Revised); Department of Computer Science, Colorado State University, Fort Collins, US

5. Fogel D. B. Evolutionary computation: towards a new philosophy of machine intelligence [Текст]/D. B. Fogel. — Piscatway: IEEE Press, 2000. — ISBN 0-7803-3481-7

6. Goldberg D. Genetic Algorithms in Search, Optimization, and Machine Learning [Текст]/ D. Goldberg. — Massachusetts: Addison-Wesley, 1989. — ISBN 0201157675

7. Hartmann A.K., Rieger H. Optimization Algorithms in Physics. — Berlin: Wiley-VCH, 2002. — 383 с. — ISBN 3527403078.

8. Holland J. H. Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence [Текст]/J. H. Holland. — The MIT Press, Cambridge, 1992. — ISBN 0262581116

9. Koza J. R. Genetic Programming [Текст]/J. R. Koza. — Cambridge: The MIT Press, 1998.— 609 с. — ISBN 0-262-11170-5

10. Michalewicz Z. Genetic algorithms + Data Structures = Evolution Programs [Текст]/Z. Michalewicz. — New York: Springer-Verlag, 1996. — 387 с. — ISBN 3540606769

11. Mitchell M. An Introduction to Genetic Algorithms [Текст]/M. Mitchell. — Cambridge: MIT Press, 1999 —158 с. — ISBN 0-262-13316-4 (HB), 0-262-63185-7 (PB)

12. Periaux J., Sefrioui M., Ganascia J.-G. Fast Convergence Thanks to Diversity // Evolutionary computing. — San Diego, 1998. — 9 с.

13. Periaux J., Sefrioui M. Evolutionary computational methods for complex design in aerodynamics // AIAA-98-0222. — Reno, 1998. — 15 с.

14. Periaux J. Combining Game Theory and Genetic Algorithms with Application to DDM-Nozzle Optimization Problems // Proceedings of DDM. — Greenwich, 1998. — 17 с.

15. Periaux J. Genetic Algorithms for electromagnetic backscattering multiobjective optimization // Genetic algorithms for Electromagnetic Computation. — Ed: Erir. Miechelssen, 1998. — 30 с.

16. Schwefel H.-P. Evolution and Optimum Seeking [Текст]/H.-P. Schwefel. — New York: John Wiley & Sons, 1995.

17. Генетические алгоритмы, Исаев А. — http://www.algolist.manual.ru

18. Генетические алгоритмы на сайте Санкт-Петербугского государственного университета информационных технологий, механики и оптики. — http://rain.ifmo.ru/cat

19. Исследования по ГА в Мичиганском университете. — http://garage.cps.msu.edu

20. Исследования по ГА в университете штата Колорадо. — http://www.cs.colostate.edu

21. Организация по Генетическим Алгоритмам. — http://www.genetic-programming.org

22. Ассоциация по Генетическим Алгоритмам университета Джорджа Мейсона. — http://www.cs.gmu.edu/research/gag

Татьяна Вячеславовна Панченко

Генетические алгоритмы

Учебное пособие
Под редакцией Ю.Ю. Тарасевича

Редактор С.Н. Лычагина
Компьютерная правка, верстка Ю.Ю. Тарасевич, Т.В. Панченко

Заказ № 1072. Тираж 50 экз.
Уч.-изд. л. 7,6. Усл. печ. л. 7,1

Издательский дом «Астраханский университет»
414056, г. Астрахань, ул. Татищева, 20
тел./ факс (8512) 54-01-89, 54-01-87, e-mail: asupress@yandex.ru