# МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Нижегородский государственный университет им. Н.И. Лобачевского**

**А.Ф. Ляхов**

**ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА**

 **К ЗАДАЧЕ АППРОКСИМАЦИИ ФУНКЦИИ**

**ПОЛИНОМОМ**

Практикум

Рекомендован методической комиссией ИТММ для студентов ННГУ,

обучающихся по направлениям подготовки

01.04.02 «Прикладная математика и информатика»

Нижний Новгород

2017

УДК 519.633

ББК В 631

 Л17

Л17 Ляхов А.Ф. Применение генетического алгоритма к задаче аппроксимации функции полиномом**:** Практикум.  Нижний Новгород: Нижегородский госуниверситет, 2017.  с.

Рецензент: д. ф.-м. н., доцент Н.Ю. Золотых

Практикум посвящен изучению одного из современных методов решения задач оптимизации − генетическому алгоритму. В качестве примера оптимизируемой задачи рассматривается задача о поиске полинома аппроксимирующего функцию.

В практикуме описаны основные этапы генетического алгоритма и их реализации в пакете Genetic Algorithm Tool (MATLAB). Предложены задания для студентов по нахождению полинома аппроксимирующего заданную функцию..

Практикум предназначен для студентов ИИТММ, обучающихся по направлению подготовки 01.04.02 «Прикладная математика и информатика». Практикум может быть полезен для обучения студентов других естественно-научных факультетов и институтов ННГУ.

УДК 519.633

ББК В.631

 **Нижегородский государственный университет**

**им. Н.И. Лобачевского, 2017**

**Содержание**

Введение……………………………………………………………………........4

1. Основные понятия и термины, используемые в генетическом алгоритме……………………………………………………………..…….4
2. Пример использования генетического алгоритма для нахождения минимума функции………………………………………………….….….6
3. Применение генетического алгоритма для задач с непрерывными параметрами………………………………………………………….…..…9
4. Реализация генетического алгоритма

в пакете Genetic Algorithm Tool (MATLAB)……………………………..……11

1. Аппроксимация функции с помощью генетического алгоритма ….…..16
2. Метод наименьших квадратов в MATLAB………………………………….…19
3. Задания ………………………………………………………………….….21

Список используемой литературы ……………………………..………….…22

**Введение**

В настоящее время при решении алгоритмически сложных задач часто используется генетический алгоритм [1− 3].

Генетический алгоритм относится к области так называемых «мягких вычислений» [1]. Заметим, что к этой области относятся методы, основанные на использовании нечётких множеств, нейронных сетей.

В основе генетического алгоритма лежат идеи, связанные с современной биологической теорией естественного отбора. В этой теории предполагается, что в процессе отбора выживают наиболее приспособленные особи с определенным набором генов, которые передаются следующим поколениям. Этот алгоритм был предложен Д. Холландом в 1975 году.

Генетический алгоритм относится к стохастическим методам решения широкого круга задач NP сложности:

- задачи численной оптимизации ;

- задачи о кратчайшем пути;

- аппроксимация функций;

- отбор (фильтрация) данных;

- настройка и обучение искусственной нейронной сети;

- игровые стратегии.

В работе описаны основные идеи генетического алгоритма. Показано, как этот алгоритм можно использовать для аппроксимации функций.

Задача аппроксимации функции является задачей оптимизации и по сути сводится к нахождению минимума некоторого функционала, описывающего расстояние между заданной функцией и функцией, которая её приближает. Все оптимизационные методы можно разделить на два класса: методы, использующие понятие производной (градиентные методы), и стохастические методы (например методы, связанные с построением нейронной сети, с нечёткими множествами и т.д.). Однако найденный с помощью этих методов экстремум может носить локальный характер. Эта проблема, как правило, решается при корректном использовании генетического алгоритма.

* + - 1. **Основные понятия и термины, используемые**

**в генетическом алгоритме**

Введем основные понятия и термины, используемые при реализации генетического алгоритма.

*Вектор* – упорядоченный набор чисел. Поскольку 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 двухэлементного множества (булева множества), например {0,1} или {-1,1}.

*Хеммингово расстояние* – используется для булевых векторов и равно числу различающихся компонент в обоих векторах. Например, для векторов

a =1011101 и b=1001001 r(a,b) = 2.

Хеммингово расстояние обладает всеми свойствами метрики.

В описании генетического алгоритма широко используется биологическая терминология.

*Хромосома* – вектор (или строка) из каких-либо чисел. Часто этот вектор представляют бинарной строкой из нулей и единиц, например 1010011. Значения вектора могут быть получены с использованием либо двоичного кодирования, либо кода Грея. Каждая позиция (бит) хромосомы называется *геном*.

*Особь –* понятие особиидентично понятию хромосома.

*Кроссинговер* (кроссовер) – операция, при которой две хромосомы обмениваются своими частями. Например, 1100&1010  1110&1000.

*Мутация* – случайное изменение одного или нескольких генов, то есть одной или нескольких позиций в хромосоме. Например, 1010011  1010001.

*Инверсия* – изменение порядка следования генов в хромосоме или в её фрагменте. Например, 1100  0011.

*Популяция* – совокупность особей.

*Пригодность* (приспособленность) – критерий или функция, экстремум которой надо найти при решении задачи.

*Локус* – разряд в бинарной строке.

Генетический алгоритм (ГА) является итерационным алгоритмом. В зависимости от решаемой задачи в качестве критерия остановки алгоритма используется достижение значений функции пригодности некоторых заданных ограничений. Поскольку вопрос о сходимости алгоритма, как правило, остается неисследованным, поэтому требуется ввести ограничения на количество допустимых итераций, то есть на количество поколений.

Основные этапы работы генетического алгоритма:

1. Генерируется начальная популяция из хромосом.
2. Вычисляется функция пригодности для каждой хромосомы.
3. Проверяется условие выполнимости критерия окончания процесса.
4. С помощью одного из способов отбора выбираются пары хромосом-родителей.
5. Проводится кроссинговер родителей, в результате которого производятся потомки.
6. Проводится операция мутации потомков.
7. В результате выполнения пунктов 3 − 6, будет сгенерировано новое поколение популяции.
	1. В новой популяции, состоящей из родителей и потомков, осуществляется операция выбора членов популяции, участвующих в дальнейшем отборе.
	2. Осуществляется переход на пункт (2).

Принципиальная блок-схема ГА показана рис. 1.



Рис. 1. Блок схема генетического алгоритма

Окончания процесса определяется схождением популяции или заданным числом поколений (числом итераций процесса).

*Схождением популяции* называется такое состояние популяции, когда все строки популяции почти одинаковы, то есть удовлетворяют заданной точности, и находятся в области некоторого экстремума.

**Операторы генетического алгоритма**

*Панмиксия* – оператор выбора родителей. В соответствии с ним каждому члену популяции сопоставляется случайное число на отрезке – количество особей в популяции. Эти числа рассматриваются 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 пары сами с собой. Какие-то члены популяции примут участие в процессе воспроизводства неоднократно с различными особями популяции. Эффективность такого алгоритма снижается с ростом численности популяции.

*Одноточечный кроссинговер* – оператор обмена генетической информации, работающий следующим образом: внутри хромосомы выбирается произвольная точка (точка разрыва), в которой хромосомы родителей делятся на две части и обмениваются ими. Например, для точки разрыва хромосомы равного 3, отсчитываются три гена и хромосомы 111|111 и 000|000 обмениваются частями, стоящими после точки разрыва, и образуют двух новых потомков: 111|000 и 000|111.

*Двоичная мутация* – случайным образом выбирается одна или несколько точек, после чего соответствующие этим точкам гены инвертируются. Вероятность мутации может являться каким-либо фиксированным числом или функцией от некоторой характеристики решаемой задачи.

Мутация необходима для “выбивания” популяции из области локального экстремума и препятствует преждевременной сходимости.

Существует несколько операторов отбора особей для нового этапа: метод усечения, отжига, вытеснения и т.д. В дальнейшем будет использоваться элитарный метод отбора.

*Элитарный отбор* – оператор отбора особей в новую популяцию. При элитарном отборе создается промежуточная популяция, которая включает в себя как родителей, так и потомков. Члены этой популяции оцениваются, а за тем из них выбираются самых лучших (пригодных), которые и войдут в следующее поколение.

1. **2. Пример использования генетического алгоритма для**
2. **нахождения минимума функции**
3. Найти глобальный минимум функции
4. $y\left(x\right)=5-24x+17x^{2}-\frac{11}{3}x^{3}+\frac{1}{4}x^{4},x\in \left[0,7\right]$.
5. На заданном отрезке функция имеет глобальный минимум в точке $x=1$, локальный минимум в точке $x$ = 6,
6. Для упрощения демонстрации применения генетического алгоритма будем полагать, что $x$ принимает лишь целые значения, то есть $x\in \left\{0,1,2,3,4,5,6,7\right\}$.
	1. **Создание популяции**

 Случайным образом выбирается несколько чисел на заданном отрезке. Пусть это будет {2,3,5,4} (рис. 2). Эти числа представляют собой особи начальной популяции.



Рис. 2

* 1. **Вычисление пригодности членов популяции**

 Приспособленность особи определяется целевой функцией: чем меньше значение целевой функции, тем более приспособленной является особь, то есть пробное решение, используется в качестве аргумента целевой функции (Таблица 1).

Таблица 1.Приспособленность особей исходного поколения

|  |
| --- |
| Особь |
| Целое число | Двоичное число | Приспособленность |
| 2 | 010 | -0,33 |
| 3 | 011 | 7,25 |
| 5 | 101 | 7,25 |
| 4 | 100 | 10,33 |

* 1. **Проверка условия выполнимости критерия окончания процесса**

В данном демонстрационном примере этот этап ГА пропускается.

* 1. **Выбор пар особей для скрещивания**

 Выбор пар осуществляется с помощью оператора панмиксия. Каждая особь исходной популяции нумеруется случайными числами в диапазоне от 1 до 4 (Таблица 2). Из этих номеров составляются случайные пары. При этом учитывается, что от каждой пары получается два потомка, и размер новой популяции должен равняться размеру исходной популяции.

Таблица 2. Панмиксия

|  |  |
| --- | --- |
| № | Особь |
| Целое число | Двоичное число |
| 1 | 2 | 010 |
| 2 | 3 | 011 |
| 3 | 5 | 101 |
| 4 | 4 | 100 |

* 1. **Процесс кроссинговера**

Процесс кроссинговера приводится в Таблице 3. Для сформированных пар выбираются произвольные точки разрыва и меняются лежащие справа от этих точек участки хромосом. В результате кроссинговера необходимо получить новую популяцию с более приспособленными особями, то есть. с более близкими к глобальному минимуму.

Таблица 3. Одноточечный кроссинговер

|  |  |  |  |
| --- | --- | --- | --- |
| Выбранныйномер | Особиродители | Точкакроссинговера | Особипотомки |
| 1 | 010 | 1 | 110 |
| 4 | 100 | 000 |
| 3 | 101 | 3 | 100 |
| 1 | 010 | 011 |

* 1. Мутация

Процедура мутации в условиях данной задачи состоит в случайном инвертировании генов хромосом, полученных после скрещивания (двоичная мутация). Предполагается, что вероятность мутации равна 0,3. Для каждого потомка берется случайное число на отрезке [0,1], и если это число меньше 0,3, то инвертируется случайно выбранный ген (замена 0 на 1 или наоборот) (Таблица 4).

Таблица 4. Мутация потомков

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Особи-потомки | Случайное число | Ген, выбранный для мутации | Потомок после мутации | Приспособленностьдо мутации | Приспособленность после мутации |
| 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 |

Мутация способна как улучшать (первый потомок), так и ухудшать (четвертый потомок) приспособленность особи-потомка. В результате скрещивания хромосомы обмениваются “хвостами”, то есть 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 особей, то мутации могут привести к существенным изменениям особей, а следовательно и решений (рис. 3).



Рис. 3. Изменение популяции в процессе естественного отбора

* 1. **Формирование новой популяции**

Для формирования новой популяции используется элитарный метод отбора. Эта процедура отражена в таблице 5.

Таблица 5. Формирование популяции элитарным методом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Особи | Приспособленность | Новая популяция | Приспособленность |
| 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 |

На рис. 4 показаны особи новой популяции .



Рис. 4. Минимизируемая функция и особи новой популяции

Получившуюся популяцию можно вновь подвергнуть кроссинговеру, мутации и отбору особей в новое поколение. В результате, если имеет место сходимость генетического алгоритма, то 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 задачи. В данном случае, взяв наиболее приспособленную особь 001 во втором поколении, можно вычислить минимум целевой функции $\begin{array}{c}y\left(\left(x\right)\right)=-5,42\\min⁡\end{array}$ , соответствующий аргументу $x=1$.

Заметим, что обычные итерационные методы поиска минимума функции могли привести к нахождению локального минимума.

1. **Применение генетического алгоритма для задач с непрерывными параметрами**

В рассмотренном примере, демонстрирующем принцип работы генетического алгоритма, решения искались среди целых чисел, а соответствующие хромосомы имели вид двоичной последовательности. Но в большинстве задач решения ищутся на множестве вещественных чисел. В таких случаях хромосома представляется вещественными генами, однако кодирование битовой строкой также возможно. Для этого вещественные хромосомы преобразуются в битовые.

*В каноническом генетическом* алгоритме хромосома представляет собой битовую строку, в которой закодированы параметры решения поставленной задачи.

Приведем один из вариантов прямого и обратного преобразования «целочисленный ген → вещественное число».

Если известен диапазон, в пределах которого лежит значение параметра, то этот диапазон разбивают на $2^{m}$отрезков, где *m* – разрядность гена, и каждому отрезку соответствует определенное значение гена:

$$g=\frac{\left(r-x\_{min}\right)\*\left(2^{m}-1\right)}{\left(x\_{max}-x\_{min}\right)}$$

для перевода значений из закодированного значения в дробные применяют следующую формулу:

$r=\frac{g\*\left(x\_{max}-x\_{min}\right)}{2^{m}-1}+x\_{min}$ ,

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

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

$r=\frac{43981\*2-1}{2^{16}-1}+1=0,6711+1=1,6711$.

Если же вещественное значение равно 1,3275, то соответствующий ген после преобразования будет выглядеть следующим образом (с округлением в меньшую сторону):

$g=\frac{\left(1,3275-1\right)\left(2^{16}-1\right)}{2-1}=0,3275\*65535=21462,7125≈21462\_{10}$= 0101 0011 1101 $0110\_{2}$.

Фактически осуществляется процедура масштабирования величины и операция квантования исходного отрезка. В таком представлении погрешность определяется величиной интервала разбиения.

Часто бывает удобнее кодировать в гене не целое число, а вещественное. Это позволяет избавиться от операций кодирования/декодирования, используемых в целочисленном кодировании, а также увеличить точность найденного решения.

В модуле Genetic Algorithm Tool (MatLab) реализованы два варианта кодирования вещественных генов. Для выбора метода во вкладке Population заполняется поле Population type. У этого поля есть выпадающий список, в котором предлагаются на выбор следующие значения для ввода: вещественный вектор, битовая строка, пользовательский тип.

При этом стоит учитывать, что использование битовой строки и пользовательских типов накладывает ограничения на перечень допустимых операторов создания, мутации и скрещивания особей.

Для вещественного кодирования стандартные операторы скрещивания и мутации не подходят, так как алгоритм работает только с вещественными числами. Поэтому были разработаны специальные операторы [4]:

− Для скрещивания: плоский кроссовер, арифметический кроссовер, BLX- кроссовер, линейный кроссовер.

− Для мутации: случайная мутация, неравномерная мутация.

Рассмотрим их более подробно.

Пусть $C\_{1}=\left(c\_{1}^{1},c\_{2}^{1},…,c\_{n}^{1}\right)$ и $C\_{2}=\left(c\_{1}^{2},c\_{2}^{2},…,c\_{n}^{2}\right)$ – две хромосомы, выбранные оператором селекции для проведения кроссовера.

Плоский кроссовер.

Создается потомок

$H=\left(h\_{1}^{},…,h\_{i}^{},…,h\_{n}^{}\right)$,

где $h\_{i},i=1,n$ – случайное число из интервала $\left[c\_{i}^{1},c\_{i}^{2}\right]$.

Арифметический кроссовер

Создаются два потомка $H\_{1}=\left(h\_{1}^{1},…,h\_{n}^{1}\right)$ и $H\_{2}=\left(h\_{1}^{2},…,h\_{n}^{2}\right)$:

$h\_{i}^{1}=ηc\_{i}^{1}+\left(1-η\right)c\_{i}^{2}$, $h\_{i}^{2}=ηc\_{i}^{2}+\left(1-η\right)c\_{i}^{1},$

$$i=´,ηϵ\left[0,1\right].$$

BLX-$α$ кроссовер.

Генерируется один потомок $H=\left(h\_{1},…h\_{i},…h\_{n}\right),$где $h\_{i}^{}$ − случайное число из интервала $\left[c\_{min}-Δ∙α,c\_{max}+Δ∙α\right]$, $c\_{max}=max⁡\left(c\_{i}^{1},c\_{i}^{2}\right)$,$c\_{min}=min\left(c\_{i}^{1},c\_{i}^{2}\right)$, $∆=c\_{max}-c\_{min}$, $i=´$.

Линейный кроссовер.

Создаются три потомка, рассчитываемые по формулам:

$h\_{i}^{1}=\frac{c\_{i}^{1}+c\_{i}^{2}}{2}, h\_{i}^{2}=\frac{3c\_{i}^{1}-c\_{i}^{2}}{2}, h\_{i}^{3}=\frac{-c\_{i}^{1}+3c\_{i}^{2}}{2}$ .

При реализации линейного кроссовера, на этапе селекции выбираются две особи с наибольшими приспособленностями.

Случайная мутация:

При случайной мутации ген, подлежащий изменению, принимает случайное значение из интервала своего изменения.

Неравномерная мутация.

В неравномерной мутации значение гена после оператора мутации рассчитывается по формуле:

$$^=\left\{\begin{array}{c}c\_{i}+δ\left(t,b\_{i}-c\_{i}\right)приμ=0\\c\_{i}-δ\left(t,b\_{i}-c\_{i}\right)приμ=1\end{array}\right.$$

$$δ\left(t,y\right)=y\left(1-r^{\left(1-\frac{t}{ε\_{max}}\right)^{b}}\right),$$

где $μ$ – целое случайное число, принимающее значение 0 или 1; $rϵ\left[0,1\right]$ – случайное вещественное число; $ε\_{max}$ – максимальное количество эпох алгоритма; *b* – задаваемый параметр.

1. **Реализация генетического алгоритма**

**в пакете Genetic Algorithm Tool (MATLAB)**

В математическом пакете Matlab существует 4 основные функции для работы с генетическим алгоритмом [1]:

**ga** – функция для нахождения минимума целевой функции;

**gaoptimget** – возвращает параметры используемого ГА;

**gaoptimset** – устанавливает параметры ГА;

**gatool** – открывает окно Genetic Algorithm Tool.

Генетический алгоритм и его 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).

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

Для запуска пакетаGenetic Algorithm Tool следует в командной строке Matlab выполнить команду **gatool**. После этого запустится пакет генетических алгоритмов и на экране появится основное окно утилиты (Рис. 5).

В поле **Fitness function** указывается оптимизируемая функция в виде **@fitnessfun**, где **fitnessfun.m** – название M-файла, в котором предварительно описывается оптимизируемая функция.

Пример описания некоторой функции **my\_fun** в M-файле:

function z = my\_fun(x)

z = x(1)^2-2\*x(1)\*x(2)+2\*x(1)+x(2)^2-3\*x(2);

end

## Рис. 5

1. В поле **Number of variables** указывается длина входного вектора оптимизируемой функции. В рассмотренном выше примере это функция **my\_fun**, которая имеет входной вектор длины 2.
2. В панели **Constraints** можно задать ограничения или ограничивающую нелинейную функцию. В поле **Linear inequalities** задается линейное ограничение неравенством вида:
3. $A\*x\leq b.$
4. В поле **Linear equalities** данной панели задаются линейные ограничения равенством:
5. $A\*x=b.$
6. В обоих случаях *A* – некоторая матрица, *b* – вектор.
7. В поле **Bounds** в векторном виде задаются нижнее и верхнее ограничения переменных, а в поле **Nonlinear constraint function** можно задать произвольную нелинейную функцию ограничений.
	1. Если в конкретной задаче задание ограничений не требуется, все поля панели Constraints остаются незаполненными.

Панель **Run Solver** содержит управляющие элементы (кнопки **Start, Pause** и **Stop** для начала, временной и полной остановки работы генетического алгоритма). Также она содержит поля **Status and results**, в которое выводятся текущие результаты работы запущенного генетического алгоритма, и **Final point**, в котором выводится значение конечной точки работы алгоритма — наилучшей величины оптимизируемой функции (то есть, искомое значение).

В правой части основного окна утилиты GATool находится панель Options (рис. 6). Она позволяет устанавливать различные настройки для работы генетических алгоритмов. При щелчке мышью по кнопкам [+], которые находятся напротив названия каждого из настраиваемых параметров в панели Options, появляются выпадающие списки (вкладки), содержащие поля для ввода и изменения соответствующих параметров генетического алгоритма.



Рис. 6

Основными параметрами в GATool являются:

* популяция (вкладка Population);
* масштабирование (вкладка Fitness Scaling);
* оператор отбора (вкладка Selection);
* оператор репродукции (вкладка Reproduction);
* оператор мутации (вкладка Mutation);
* оператор скрещивания (вкладка Crossover);
* перенесение особей между подпопуляциями (вкладка Migration);
* специальные параметры алгоритма (вкладка Algorithm settings);
* задание гибридной функции (вкладка Hybrid function);
* задание критерия остановки алгоритма (вкладка Stopping criteria);
* вывод различной дополнительной информации по ходу работы генетического алгоритма (вкладка Plot Functions);
* вывод результатов работы алгоритма в виде новой функции (вкладка Output function);
* задание набора информации для вывода в командное окно (вкладка Display to command window);
* способ вычисления значений оптимизируемой и ограничивающей функций (вкладка User function evaluation).

**Population**

Вкладка популяции позволяет настраивать размер популяции (из скольких особей будет состоять каждое поколение) и определить каким образом будет создаваться начальное поколение (Uniform – если отсутствуют налагаемые ограничения, в противном случае — Feasible population). Кроме того, в рассматриваемой вкладке имеется возможность задать вручную начальное поколение (используя пункт Initial population) или его часть, начальный рейтинг особей (пункт Initial scores), а также ввести ограничительный числовой диапазон, которому должны принадлежать особи начальной популяции (Initial range).

**Fitness Scaling**

Во вкладке масштабирования (Fitness Scaling) пользователь имеет возможность указать функцию масштабирования, которая конвертирует достигаемые оптимизируемой функцией значения в значения, лежащие в пределах, допустимых для оператора отбора. При выборе в качестве функции масштабирования параметра Rank масштабирование будет приводиться к рейтингу, то есть особям присваивается рейтинговый номер (для лучшей особи – единица, для следующей – двойка и так далее). Пропорциональное масштабирование (Proportional) задает вероятности пропорционально заданному числовому ряду для особей. При выборе опции Top наибольшее рейтинговое значение присваивается сразу нескольким наиболее выдающимся особям (их число указывается в виде параметра). Наконец, при выборе масштабирования типа Shift linear имеется возможность указать максимальную вероятность наилучшей особи.

**Selection**

Вкладка Selection позволяет выбрать оператор отбора родительских особей на основе данных из функции масштабирования. В качестве доступных для выбора вариантов оператора отбора предлагаются следующие:

* Tournament – случайно выбирается указанное число особей, среди них на конкурсной основе выбираются лучшие;
* Roulette – имитируется рулетка, в которой размер каждого сегмента устанавливается в соответствии с его вероятностью;
* Uniform – родители выбираются случайным образом согласно заданному распределению и с учетом количества родительских особей и их вероятностей;
* Stochastic uniform – строится линия, в которой каждому родителю ставится в соответствие её часть определенного размера (в зависимости от вероятности родителя), затем алгоритм пробегает по линии шагами одинаковой длины и выбирает родителей в зависимости от того, на какую часть линии попал шаг.

**Reproduction**

Вкладка Reproduction уточняет каким образом происходит создание новых особей. Пункт Elite count позволяет указать число особей, которые гарантировано перейдут в следующее поколение. Пункт Crossover fraction указывает долю особей, которые создаются путем скрещивания. Остальная доля создается путем мутации.

**Mutation**

Во вкладке оператора мутации выбирается тип оператора мутации. Доступны следующие варианты:

* Gaussian добавляет небольшое случайное число (согласно распределению Гаусса) ко всем компонентам каждого вектора-особи;
* Uniform выбираются случайным образом компоненты векторов и вместо них записываются случайные числа из допустимого диапазона;
* Adaptive feasible генерирует набор направлений в зависимости от последних наиболее удачных и неудачных поколений и с учетом налагаемых ограничений продвигается вдоль всех направлений на разную длину;
* Custom позволяет задать собственную функцию.

**Crossover**

Вкладка Crossover позволяет выбрать тип оператора скрещивания (одноточечное, двухточечное, эвристическое, арифметическое или рассеянное (Scattered), при котором генерируется случайный двоичный вектор соответствия родителей). Также имеется возможность задания произвольной (custom) функции скрещивания.

**Migration**

Во вкладке Migration можно настраивать правила, согласно которым особи будут перемещаться между подпопуляциями в пределах одной популяции. Подпопуляции создаются, если в качестве размера популяции указан вектор, а не натуральное значение. В данной вкладке можно указать направление миграции (forward – в следующую подпопуляцию, both – в предыдущую и следующую), долю мигрирующих особей и частоту миграции (сколько поколений проходит между миграциями). Если создание подпопуляций не требуется, эту вкладку всегда стоит оставлять без изменений.

**Algorithm settings**

Вкладка специальных опций алгоритма позволяет настраивать параметры решения системы нелинейных ограничений, налагаемых на алгоритм. Значение параметра Initial penalty определяет начальное числовое значение критики алгоритма, Penalty factor используется как множитель этого значения в случаях, когда разработчика не устраивает точность оптимизации или при выходе за границы, определенные во вкладке ограничений. Как правило, эти опции детально настраиваются для решения задач высокой сложности.

**Hybrid function**

Вкладка Hybrid function позволяет задать ещё одну функцию минимизации, которая будет использоваться после окончания работы алгоритма. В качестве возможных гибридных функций доступны следующие встроенные в саму среду MATLAB функции:

* none (не использовать гибридную функцию);
* fminsearch (поиск минимального из значений);
* patternsearch (поиск по образцу);
* fminunc (для неограниченного алгоритма);
* fmincon (для алгоритма с заданными ограничениями).

**Stopping criteria**

Во вкладке критерия остановки (Stopping criteria) указываются ситуации, при которых алгоритм совершает остановку. При этом, настраиваемыми являются следующие параметры:

* Generations – максимальное число поколений, после превышения которого произойдет остановка;
* Time limit – лимит времени на работу алгоритма;
* Fitness limit – если оптимизируемое значение меньше или равно заданному ограничению, то алгоритм остановится;
* Stall generations – количество малоотличающихся поколений, по прошествии которых алгоритм остановится;
* Stall time limit – то же, что и предыдущий параметр, но применимо к времени работы алгоритма;
* Function tolerance и Nonlinear constraint tolerance – соответственно минимальные значения изменений оптимизируемой и ограничивающей функций, при которых алгоритм продолжит работу.

**Plot Functions**

Вкладка Plot Functions позволяет выбирать различную информацию, которая выводится по ходу работы алгоритма и показывает как корректность его работы, так и конкретные достигаемые алгоритмом результаты. Наиболее важными и используемыми для отображения параметрами являются:

* Plot interval – количество поколений, по прошествии которого происходит очередное обновление графиков;
* Best fitness – вывод наилучшего значения оптимизируемой функции для каждого поколения;
* Best individual – вывод наилучшего представителя поколения при наилучшем оптимизационном результате в каждом из поколений;
* Distance – вывод интервала между значениями особей в поколении;
* Expectation – выводит ряд вероятностей и соответствующие им особи поколений;
* Genealogy – вывод генеалогического дерева особей;
* Range – вывод наименьшего, наибольшего и среднего значений оптимизируемой функции для каждого поколения;
* Score diversity – вывод гистаграммы рейтинга в каждом поколении;
* Scores – вывод рейтинга каждой особи в поколении;
* Selection – вывод гистограммы родителей;
* Stopping – вывод информации о состоянии всех параметров, влияющих на критерии остановки;
* Custom – отображение на графике некоторой указанной пользователем функции.

**Output function**

Вкладка вывода результатов в виде новой функции (Output function) позволяет включить вывод истории работы алгоритма в отдельном окне с заданным интервалом поколений (флаг History to new window и поле Interval соответственно), а также позволяет задать и вывести произвольную выходную функцию, задаваемую в поле Custom function.

Display to command window

Вкладка Display to command window позволяет настраивать информацию, которая отображается в основном командном окне MATLAB при работе алгоритма. Возможны следующие значения: Off — нет вывода в командное окно, Iterative — вывод информации о каждой итерации работающего алгоритма, Diagnose — вывод информации о каждой итерации и дополнительных сведениях о возможных ошибках и измененных ключевых параметрах алгоритма, Final – выводится только причина остановки и конечное значение.

User function evaluation

Вкладка User function evaluation описывает, в каком порядке происходит вычисление значений оптимизируемой и ограничивающей функций (отдельно, параллельно в одном вызове или одновременно).

1. **Аппроксимация функции с помощью генетического алгоритма**

Задача аппроксимации функции полиномами является одной из основных задач численных методов. Существует множество классических методов аппроксимации функций. Развитие современных информационных технологий позволило предложить новые методы: аппроксимация функций с помощью нейронных сетей, с помощью нечетких множеств и с помощью ГА.

 Задача аппроксимации ставится следующим образом.

Дано множество точек $\{x\_{i},i=1,…,n\}$ и известны значения некоторой функции $\{y\_{i},i=1,…,n\}$ в этих точках. Требуется найти приближающую функцию $p\left(x\right)$, такую что $p\left(x\_{i},a\right)=f\left(x\_{i}\right)$, $i=1,…n$ , где $a=\{a\_{1},…,a\_{m}\}$ – параметры приближающей функции.

Чтобы решить эту задачу с помощью генетического алгоритма, её нужно сформулировать как задачу оптимизации некоторой целевой функции. В качестве такой целевой функции может выступать норма разности заданной функции в точках и построенной аппроксимирующей функции.

Пример

С помощью генетического алгоритма построить полином третьей степени аппроксимирующий функцию

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

Создадим соответствующий М-файл:

function y = f(x)

y = (-12\*(x+4).^(2/3)+8\*x-16);

В качестве аппроксимирующей функции возьмем полином третьей степени

$P\left(x\right)=k\left(1\right)x^{3}+k\left(2\right)x^{2}+k\left(3\right)x+k\left(4\right)D$,

где $k\left(1\right),k\left(2\right),k\left(3\right),k\left(4\right)$ – неизвестные коэффициенты.

Создадим М-функцию:

function y = p(x,k)

y = k(1)\*x.^3+k(2)\*x.^2+k(3)\*x+k(4);

Целевая функция имеет вид $F\left(x\right)=\sum\_{i=1}^{n}\left(f\left(x\_{i}\right)-P\left(x\_{i}\right)\right)^{2}$

Соответствующая М-функция:

function y = fitnes(kft)

x =-4:.1:4; %узлы, в которых вычисляются значения функции

z = (f(x)-p(x,kft));

y = norm(z).^2;

В командном окне вводится команда gatool, после чего на экране появится toolbox для работы с генетическими алгоритмами.

В полях Fitness function и Number of variables вводится соответственно @fitnes и 4. В остальных полях задаются следующие параметры работы генетического алгоритма:

Population type – double vector (тип данных в популяции – вещественный вектор) ;

Population size – specify: 80 (размер популяции);

Initial Range – specify: [-4;4] (матрица или вектор, определяющие особи в начальной популяции);

Elite Count – specify: 1 (число, определяющее количество особей в текущем поколении, которые будут скопированы в новое поколение);

Generations – specify: 200 (максимальное число итераций алгоритма);

Stall generations – specify: 100 (алгоритм остановится, если за данное число последовательных итераций не произойдет улучшения функции цели ).

Значения остальных полей остаются по умолчанию.

Для графической иллюстрации работы алгоритма в полях вкладки Plot functions ставятся флажки в пунктах: Best fitness, Best individual, Distance .

Алгоритм запускается кнопкой Start панели Run Solver, после чего на экран выводится окно с выбранными графиками (рис. 7 −9).



Рис. 7



Рис. 8



Рис. 9

После завершения работы алгоритма в поля Status and results и Final point панели Run Solver выводятся значение целевой функции и найденное решение (рис. 10).



Рис. 10

Полученные коэффициенты подставляются в искомый полином и выводятся графики (рис. 11).

koft = [-0.1, 0.227, 3.684, -45.758];% полученные коэффициенты

x = -4:.1:4;

y1 = (-12\*(x+4).^(2/3)+8\*x-16);%аппроксимируемая ф-ия

y2 = koft(1)\*x.^3+koft(2)\*x.^2+koft(3)\*x+koft(4);%полином

figure(3);

plot(x,y1,'b',x,y2,'r')

legend('аппроксимируемая ф-ия','аппроксимирующий полином')



Рис. 11

Сравним аппроксимационный полином, построенный с помощью ГА, с полиномом, полученным методом наименьших квадратов.

1. **Метод наименьших квадратов в MATLAB**

Одним из способов приближения данных некоторой непрерывной функцией является приближение полиномом по методу наименьших квадратов.

 Для набора данных:

$$\left(x\_{i},y\_{i}\right)\_{i=1,2,…,N}$$

требуется найти такой полином степени *n*

$p^{\left(n\right)}\left(x\right)=p\_{1}x^{n}+p\_{2}x^{n-1}+…p\_{n}x+p\_{n+1}$,

коэффициенты которого являются решением следующей задачи минимизации:

$\min\_{p\_{1},p\_{2},…,p\_{n+1}}\sum\_{i=1}^{N}\left(p^{\left(n\right)}\left(x\_{i}\right)-y\_{i}\right)^{2}$,

то есть разыскивается полином, который наименее уклоняется от заданных значений в том смысле, что сумма квадратов расстояний от заданных точек $\left(x\_{i},y\_{i}\right)$ до $\left(x\_{i},p^{n}\left(x\_{i}\right)\right)$ будет минимальной.

В MATLAB для приближения данных в смысле наименьших квадратов используется функция polyfit, во входных аргументах которой указываются вектора с данными, а выходным является вектор коэффициентов полинома, начиная со старшей степени. Функция polyfit может быть вызвана и с большим числом входных и выходных аргументов, если это потребуется для улучшения качества приближения и получения некоторой дополнительной информации о нем.

x = -4:.1:4;

y = f(x);

[p1,info] = polyfit(x,y,3); % в p1 заносятся полученные коэффициенты

y1 = p(x,p1); % значения полученного полинома

figure(7);

plot(x,y,x,y1,'r')

legend('ф-ия f(x)','полином P(x)')

Полученный полином представлен на рис. 12.



Рис. 12

Для того чтобы узнать, насколько сильно отклоняется полученный полином от заданных точек, то есть какая была допущена погрешность при приближении данных, следует вызывать функцию polyfit с двумя выходными аргументами. В первый из них запишется вектор коэффициентов построенного полинома, а во второй − структура с информацией о приближении:

[p1,info] = polyfit(x,y,3); % в p1 заносятся полученные коэффициенты

info.normr % норма полинома

В итоге на экран будет выведена норма разности аппроксимируемой функции и полученного полинома:

ans = 3.3271.

Чтобы вычислить отклонение полинома, полученного с помощью генетического алгоритма, используется функция norm, которая высчитывает норму вектора. В качестве аргумента для этой функции берется разница между аппроксимируемой функцией и полиномом:

z = (f(x)-p(x,kft));

y = norm(z)

В результате получим:

ans = 7.3314.

Сравним время работы методов.

**Время работы МНК:**

tic;

 [p1,info] = polyfit(x,y,3); % в p1 заносятся полученные коэффициенты

toc;

На экран буде выведено:

Elapsed time is 0.005374 seconds.

**Время работы ГА:**

tic;

 ga(@fitnes);

toc;

На экран будет выведено:

Elapsed time is 3.104213 seconds.

Можно видеть, что в данном примере метод наименьших квадратов приближает исходную функцию лучше, чем генетический алгоритм. По времени работы генетический алгоритм также уступает методу наименьших квадратов.

На основе рассмотренного примера можно сделать вывод, что применение базовых методов к решению оптимизационных задач низкой степени сложности является более эффективным по сравнению со стохастическими методами, к которым относится генетический алгоритм. Однако в задачах, где применение базовых методов является затруднительным, использование генетического алгоритма более целесообразно.

1. **Задания**

1. Аппроксимировать функцию с помощью генетического алгоритма кубическим полиномом.

1. $y=x^{2}-5∙x+4∙sin⁡\left(x\right)$, $x\in \left[0;5\right]$,
2. $y=x^{2}-5∙ln⁡\left(x\right)+4∙sin⁡\left(x\right)$, $x\in \left[0;5\right]$,
3. $y=x^{2}-5∙x-4∙sin⁡\left(x\right)$, $x\in \left[0;5\right]$,
4. $y=x-5∙ln⁡\left(x\right)+4∙sin⁡\left(x\right)$, $x\in \left[0;5\right]$,
5. $y=x+4∙sin\left(x\right)-5∙sin\left(2x\right),x\in \left[0;5\right]$,
6. $y=x+4∙ln⁡\left(x\right)-5∙sin\left(2x\right),x\in \left[0;5\right]$,
7. $y=x+4∙sin\left(x\right)-5∙sin\left(2x\right)+sin⁡\left(3x\right),x\in \left[0;5\right]$,
8. $y=x+4∙ln\left(x\right)-5∙sin\left(2x\right)+sin⁡\left(3x\right),x\in \left[0;5\right]$,
9. $y=x+4∙sin\left(x\right)-5∙ln\left(2x\right),x\in \left[0;5\right]$,
10. $y=x+4∙sin\left(x\right)-5∙sin\left(2x\right),x\in \left[0;5\right]$,
11. Построить генетический алгоритм с различными параметрами и сравнить между собой результаты и скорость сходимости.
12. Аппроксимировать функцию методом наименьших квадратов и сравнить с предыдущей аппроксимацией.

**Список используемой литературы**

1. Панченко Т.В. Генетические алгоритмы. Астрахань: Издательский дом «Астраханский университет», 2007.- 87 c.
2. Гладков Л.А., Курейчик В.В., Курейчик В. М. Генетические алгоритмы / Под ред. В. М. Курейчика. 2-е изд., испр. и доп. М.: ФИЗМАТЛИТ, 2006.- 320 с.
3. Батищев Д.И. Генетические алгоритмы решения экстремальных задач. Нижний Новгород: Изд-во Нижегородского госуниверситета, 1995.- 62с.

Паклин Н.Б. Генетические алгоритмы с вещественным кодированием. URL: <http://paklin.newmail.ru/mater/rcga.html>

Александр Федорович **Ляхов**

**ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА**

 **К ЗАДАЧЕ АППРОКСИМАЦИИ ФУНКЦИИ ПОЛИНОМОМ**

***Практикум***

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского».

603950, Нижний Новгород, пр. Гагарина, 23.