Научная статья
Original article
УДК 004.021

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ ЗАДАЧИ ОПТИМИЗАЦИИ МУЛЬТИМОДАЛЬНОЙ ФУНКЦИИ

GENETIC ALGORITHMS FOR THE OPTIMIZATION PROBLEM OF A MULTIMODAL FUNCTION

Душина Виктория Александровна, студент Балтийский федеральный университет Им. И. Канта (236041 Россия, г. Калиниград, ул. Александра невского, д. 14), тел. 8(983)268-82-59, vktr_dshn@mail.ru

Dushina Victoria Alexandrovna, student of the Baltic Federal University. I. Kant (236041 Russia, Kaliningrad, Alexander Nevsky St., 14), tel. 8(983)268-82-59, vktr_dshn@mail.ru

Аннотация. Эволюционные алгоритмы представляют собой самоадаптивные методы параллельного поиска на основе популяций. Целью данной работы является исследование работы генетического алгоритма, влияния входных параметров и разных способов работы с популяциями, хромосомами, особями на результат работы, на вычисляемые переменные и значение целевой функции.

Abstract. Evolutionary algorithms are self-adaptive population-based parallel search methods. The purpose of this work is to study the operation of a genetic algorithm, the influence of input parameters and different ways of working with

populations, chromosomes, individuals on the result of the work, on the calculated variables and the value of the objective function.

Ключевые слова. Генетический алгоритм; Matlab; конфигурации; мутации; битовые строки; утилиты; операторы выбора; масштабирование.

Keywords. Genetic algorithm; matlab; configuration; mutations; bit strings; utilities; selection operators; scaling.

Введение

Генетический алгоритм (ГА) - это вид эволюционных алгоритмов, основанный на человеческой наследственности хромосомной информации, и имеет возможность определять глобальные оптимальные решения.

Генетический алгоритм является одним из наиболее авторитетных эволюционных алгоритмов, основанных на человеческой наследственности хромосом. Идея генетического алгоритма вдохновлена дарвиновской теорией эволюции «выживают наиболее приспособленные». Он порождает глобальные или близкие к глобальным оптимумам решения проблемы минимизации, создавая несколько групп особей по целому ряду итераций. ГА имеет четыре этапа, такие как инициализация, воспроизведение, кроссовер и мутация.

Исходная совокупность кандидатов решений генерируется случайным образом. Приближенность к глобальному оптимуму каждой новой особи оценивается по ее пригодности, которая может быть определена функцией пригодности. После того, как особи выбраны в качестве родителей для размножения, они производят потомство через процессы кроссовера и мутации. Потомки, сформированные во время воспроизведения, исследуют различные области пространства решений.

1. Теоретическая часть

Генетические алгоритмы оперируют совокупностью особей (популяцией), которые представляют собой строки, кодирующие одно из

решений задачи. Этим ГА отличается от большинства других алгоритмов оптимизации, которые оперируют лишь с одним решением, улучшая его. С помощью функции приспособленности среди всех особей популяции выделяют наиболее приспособленных (более подходящие решения), которые получают возможность скрещиваться и давать потомство и наихудших (плохие решения), которые удаляются из популяции и не дают потомства. Таким образом, приспособленность нового поколения в среднем выше предыдущего.

В классическом ГА:

- начальная популяция формируется случайным образом;
- размер популяции (количество особей N) фиксируется и не изменяется в течение работы всего алгоритма;
- каждая особь генерируется как случайная L-битная строка, где L — длина кодировки особи;
- длина кодировки для всех особей одинакова.

Шаг алгоритма состоит из трех стадий:

1. генерация промежуточной популяции (intermediate generation) путем отбора (selection) текущего поколения.
2. скрещивание (recombination) особей промежуточной популяции путем кроссовера (crossover), что приводит к формированию нового поколения.
3. мутация нового поколения.

Изменение параметров алгоритма может определять значения входных параметров для оптимальной работы алгоритма. Генетический алгоритм (ГА), подобно другим метаэвристикам, является эволюционным демографическим алгоритмом. То есть, популяция решений развивается в ходе оптимизации, приближаясь к оптимальному значению. Затем популяция будет сортироваться в зависимости от пригодности особи.

Кроссовер и мутация являются двумя очень важными операторами алгоритма. В обоих случаях важную роль играет функция выбора. Кроссовер

является бихромосомным оператором в том смысле, что он работает на двух хромосомах для вывода других. Две входящие хромосомы смешиваются и вырабатывается одна или две новые хромосомы, которые известны как потомки. В случае бинарной хромосомы наиболее широко используется одноточечный кроссовер. В одном шаге кроссовера обе хромосомы разбиваются с одного и того же номера клетки (гена), и их части меняются между ними, в результате чего образуются две разные хромосомы, которые имеют характеристики входящих хромосом. Кроссовер полезен тем, что он повышает оптимальность, формируя лучшие решения.

Мутация, в отличие от кроссовера, не является бихромосомной и не служит цели приближения особей к оптимальности/пригодности. Ее вклад в алгоритм состоит в том, чтобы избежать локальных экстремумов целевой функции путем радикального изменения входящих хромосом. Одиночная входящая хромосома изменяется оператором жестко, без каких-либо причин и случайным образом.

Важным этапом в алгоритме является определение момента, когда нужно остановить эволюцию и удовлетвориться лучшим ответом на данном шаге. ГА не может быть на 100% точен в оптимальном решении. Фактически, существуют разные критерии остановки.

1.1. Отбор

Промежуточная популяция — это набор особей, получивших право размножаться. Наиболее приспособленные особи могут быть записаны туда несколько раз, наименее приспособленные с большой вероятностью туда вообще не попадут.

В классическом ГА вероятность каждой особи попасть в промежуточную популяцию пропорциональна ее приспособленности, т.е. работает пропорциональный отбор.

Скрещивание

Особи промежуточной популяции случайным образом разбиваются на пары, потом с некоторой вероятностью скрещиваются, в результате чего получаются два потомка, которые записываются в новое поколение, или не скрещиваются, тогда в новое поколение записывается сама пара.

В классическом ГА применяется одноточечный оператор кроссовера: для родительских строк случайным образом выбирается точка раздела, потомки получаются путём обмена отсечёнными частями.

Мутация

К полученному в результате отбора и скрещивания новому поколению применяется оператор мутации, необходимый для "выбивания" популяции из локального экстремума и способствующий защите от преждевременной сходимости.

Можно выбирать некоторое количество точек в хромосоме для инверсии, причем их число также может быть случайным. Также можно инвертировать сразу некоторую группу подряд идущих точек.

2. Практическая часть

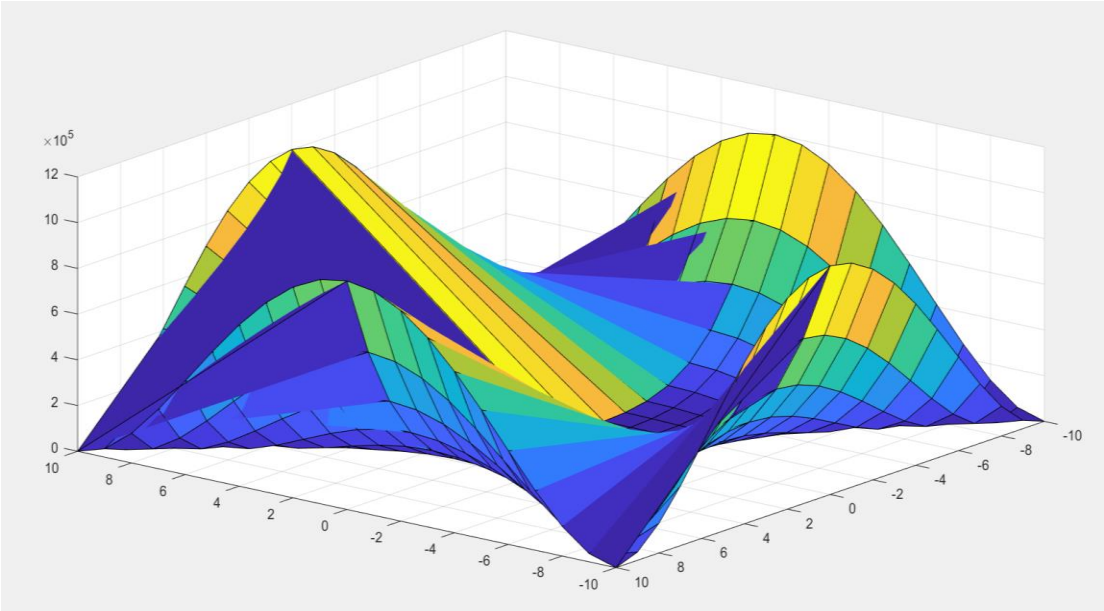
2.1. Среда и исходные данные

В качестве среды для экспериментов была выбрана программа Matlab с утилитой GATool. В данной утилите можно указать необходимые данные, выбрать конфигурацию генетического алгоритма и запустить его, получив результат. Для проверки работы генетического алгоритма была выбрана целевая функция, указанная в формуле 1.

$$f(x) = 100(x_1^2 - x_2^2)^2 + (1 - x_1)^2, \quad (1)$$

где -2,048 $\leq x_i \leq$2,048

В данной области поиска глобальный минимум функции находится в точке $x_1=1$, $x_2=1$, при этом значение функции равно 0. То есть показателем эффективности работы ГА будет максимальное приближение результата к этим значениям.



2.2. Тестирование

Для запуска пакета Genetic Algorithm Tool следует в командной строке MATLAB выполнить команду Optimtool. После этого запустится пакет генетических алгоритмов и на экране появится основное окно утилиты (Рисунок 1).

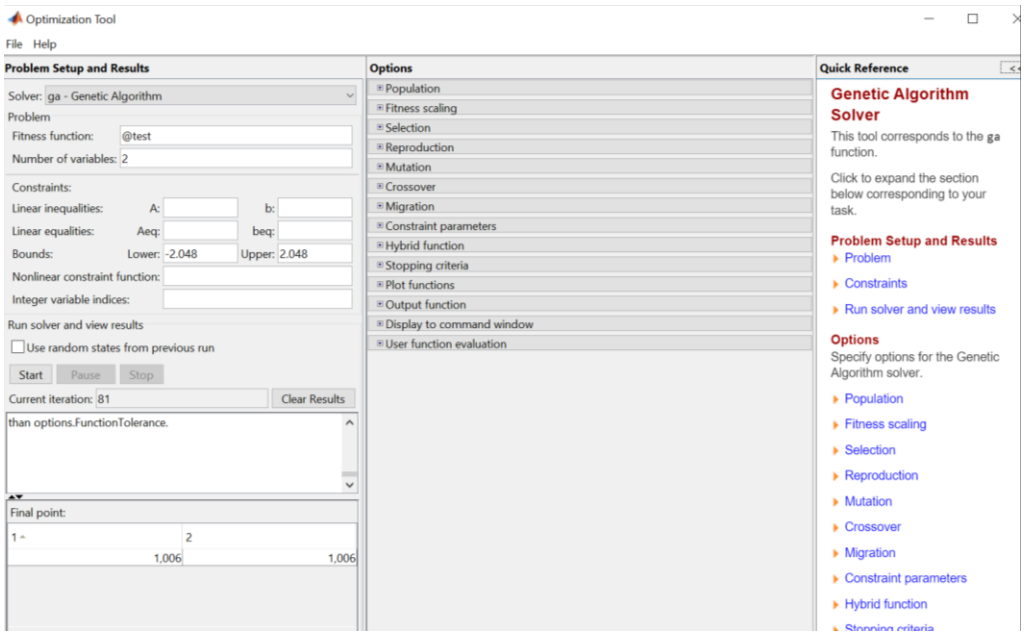


Рисунок 1 – Основное окно утилиты GATool

Напишем М-файл для данной функции и сохраним его под именем test.m (рисунок 2).

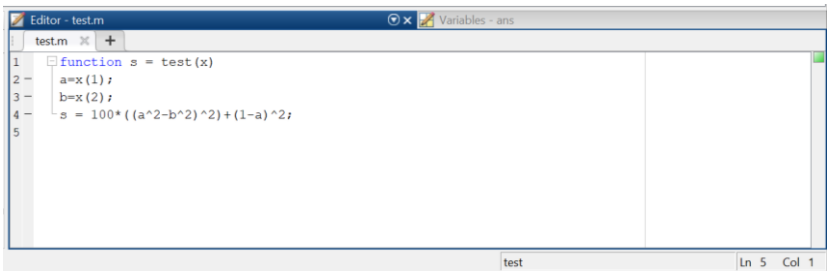


Рисунок 2 – М-файл для мультимодальной функции

2.2.1. Настройка поколений, критерий остановки

Во вкладке настройки популяций можно выбрать тип математических объектов, к которому будут относиться особи всех популяций (двойной вектор, битовая строка или пользовательский тип). При этом стоит учитывать, что использование битовой строки и пользовательских типов накладывают ограничения на перечень допустимых операторов создания, мутации и скрещивания особей. Так, например, при выборе в качестве формы представления особей битовой строки для оператора скрещивания нельзя использовать гибридную функцию или нелинейную ограничивающую функцию.

— Generations – максимальное число поколений, после превышения которого произойдет остановка. Чем меньше поколений будет сформировано генетическим алгоритмом, тем меньше вероятность получить верный результат. Однако и огромное число поколений не гарантирует лучшего результата. На рисунке 3 показан результат работы ГА, где количество поколений равно 100.

Результаты тестирования представлены в таблице 1.

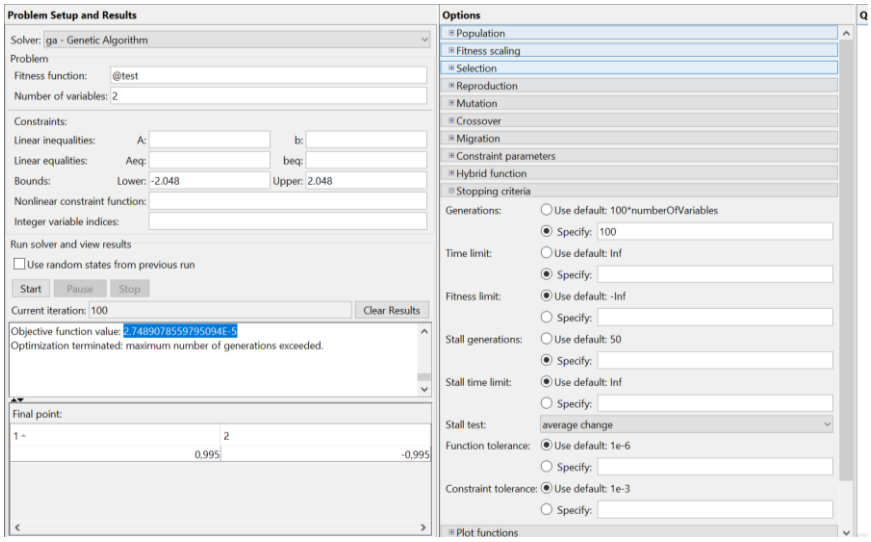


Рисунок 3 – Результат тестирования, где количество поколений 100

Таблица 1 – Критерий остановки – количество поколений

Количество поколений	x_1	x_2	Значение целевой функции
100	0,995	-0,995	2.7489078559795094E-5
200	0,967	0,966	0,0012792321245089036
300	1	1	3,670340066E-9
500	0,987	0,987	1,793882254E-4

— Time limit – лимит времени на работу алгоритма. В среде Matlab генетические алгоритмы выполняются довольно быстро и лимит времени на них не имеет значения, если он не близок к долям секунды. На рисунке 4 показан результат работы ГА с лимитом времени равным 10 секундам:

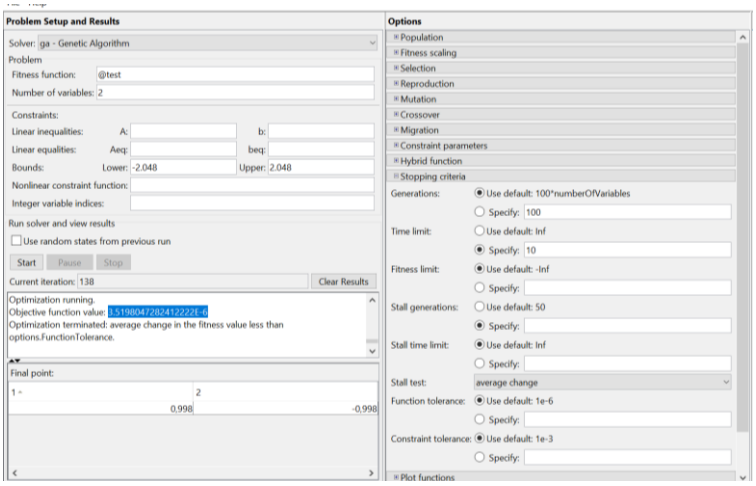


Рисунок 4 – Результат тестирования при 10 секундном лимите

В таблице 2 показаны результаты работы генетического алгоритма с установленным данным критерием остановки.

Таблица 2 – Критерий остановки – лимит времени

Лимит времени, сек	x_1	x_2	Значение целевой функции
10	0,998	-0,998	3.5198047282412222E-6
0,5	1,022	-1,022	4,788096775E-4
0,2	0,955	0,955	0,002004255196642691
0,1	0,829	0,829	0,029263808973

— Fitness limit – если оптимизируемое значение меньше или равно данного лимита, то алгоритм остановится. По умолчанию этот параметр равен – Inf, и, если поставить его больше нуля, алгоритм будет останавливаться после 1 или 2 поколения, так как исходная функция имеет глобальный минимум при значении 0. Однако при значениях меньших или равных 0 данный критерий не влияет на работу ГА. На рисунке 5 показан результат работы ГА при заданном лимите функции:

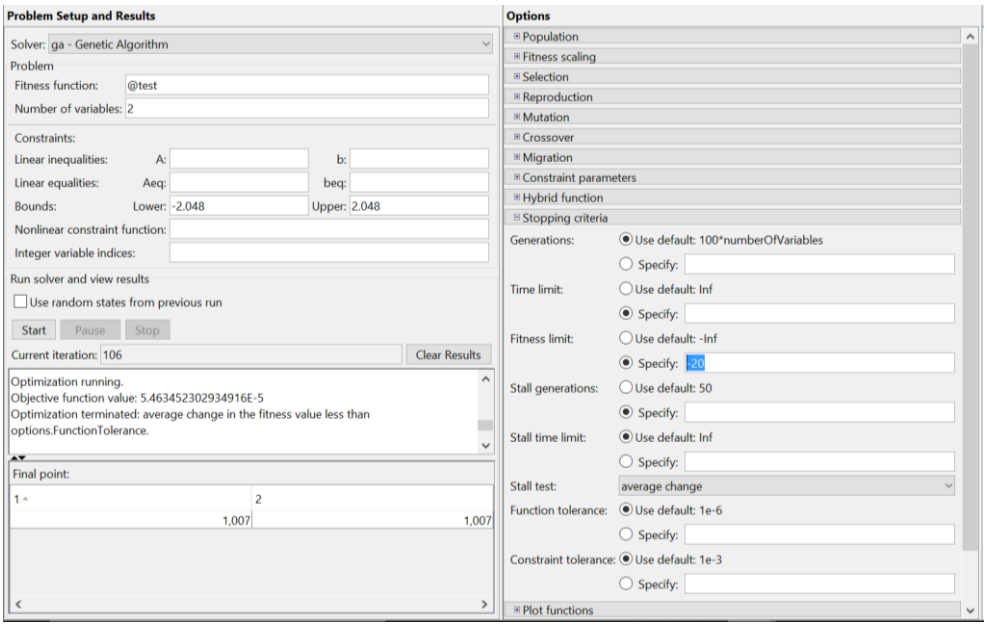


Рисунок 5 – Результат тестирования при лимите функции равном -20.

Результаты тестирования представлены в таблице 3.

Таблица 3 – Критерий остановки – лимит оптимизируемой функции

Лимит функции	x_1	x_2	Значение целевой функции
-Inf	0,929	0,929	0,005065643769753
1	0,556	0,557	0,19734853006726943
0	0,958	0,958	0,001784038860709
-20	1,007	1,007	5.463452302934916E-5

— Stall generations и Stall time limit – количество мало отличающихся поколений (второй применимо к времени работы), по прошествии которых алгоритм остановится. Данные параметры также будут влиять на работу ГА в том случае, когда их значения находятся в интервале (0;10). Результат работы ГА показан на рисунке 6, при Stall generations = 10 и Stall time limit = Inf.

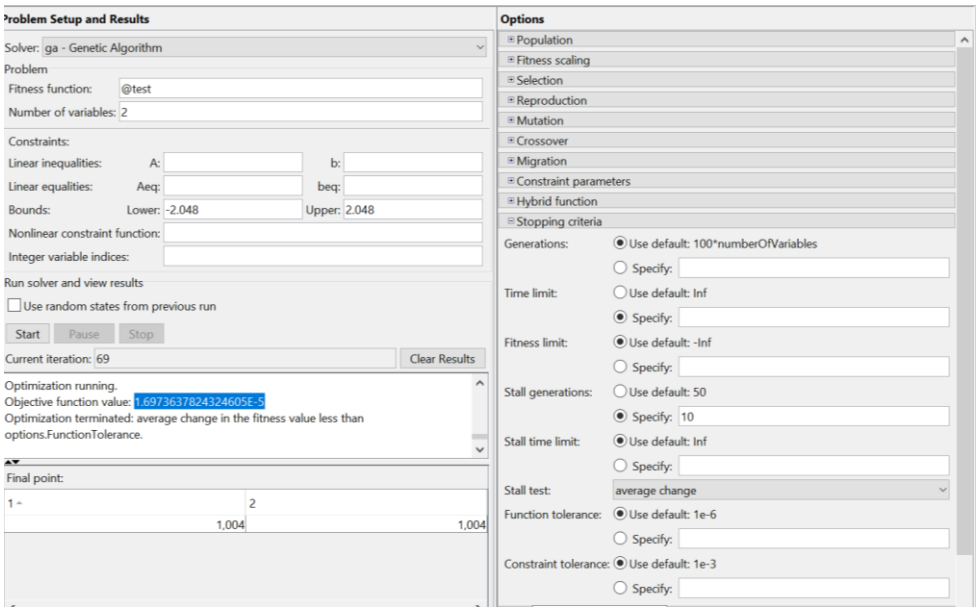


Рисунок 6 – Результат тестирования при Stall generations = 10 и Stall time limit = Inf

Результаты тестирования представлены в таблице 4.

Таблица 4 – Критерий остановки – Количество мало отличающихся поколений

Поколения/время, сек	x_1	x_2	Значение целевой функции
10/Inf	1,004	1,004	1.6973637824324605E-5
5/Inf	0,848	0,87	0,16917386897085
2/Inf	0,579	0,576	0,1782642789373
50/10	0,917	0,917	0,00690602660482
50/5	0,999	-0,999	5,572872606974E-7
50/2	1,025	1,025	6,2622216379E-4

— Function tolerance и Nonlinear constraint tolerance – минимальные значения изменений оптимизируемой и ограничивающей функций соответственно, при которых алгоритм продолжит работу. Изменение этих параметров в ходе работы не показало разницы в результатах.

— Population Size (размер популяции) (из скольких особей будет состоять каждое поколение) и каким образом будет создаваться начальное поколение (Uniform – если отсутствуют накладываемые ограничения, в противном случае — Feasible population).

Для того, чтобы все функции работали без ограничений, в экспериментах использовался тип двойной вектор. По умолчанию для функции 2 переменных размер популяции 50 особей. При увеличении количества особей в популяции точность расчетов увеличивается. Однако большое количество особей не гарантирует лучший результат. На рисунке 7 показан результат тестирования ГА при размере популяции равном 100:

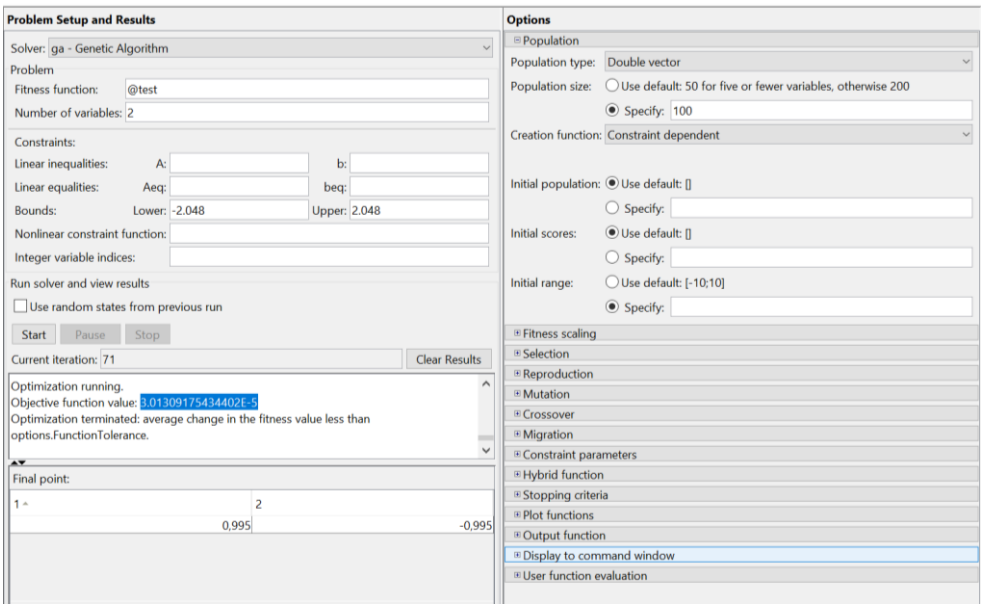


Рисунок 6 – Результат тестирования при размере популяции равном 100

Результаты представлены в таблице 5.

Таблица 5 – Настройки популяции – Размер популяции

Размер популяции, особей	x_1	x_2	Значение целевой функции
50	0,894	-0,894	0,011132398116464366
100	0,995	0,995	3.01309175434402E-5
100	0,999	0,999	3,926670723500969E-7
200	1,003	1,003	9,39646866527E-6
300	1	1	1,810043466159E-9
400	1	1	1,529618009632E-9

2.2.2. Оператор отбора (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, затем алгоритм пробегает линии шагами одинаковой длины и выбирает родителей в зависимости от того, на какую часть линии попал шаг.
- Remainder – назначает родителей детерминированно из целочисленной части масштабируемого значения каждой особи, а затем использует Roulette в оставшейся дробной части.

На рисунке 8 показан результат работы ГА с remainder оператором отбора:

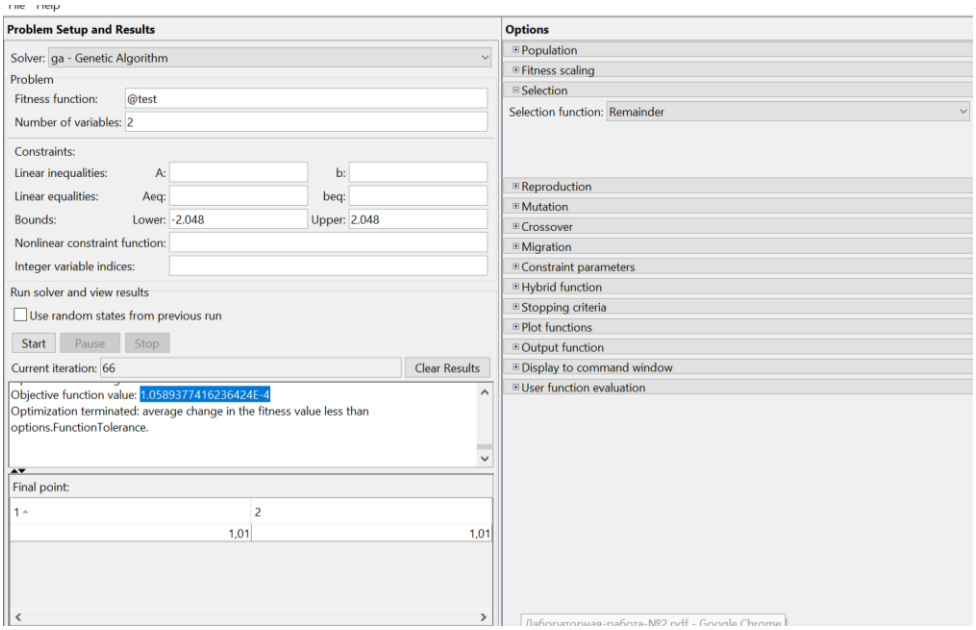


Рисунок 8 – Результат работы remainder-оператора

Результат тестирования разных операторов отбора для работы ГА представлен в таблице 6.

Таблица 6 – Операторы отбора

Функция масштабирования	x_1	x_2	Значение целевой функции
tournament	0,92	-0,92	0,0063887844902
tournament	1,013	1,013	1,661844422513
remainder	1,01	1,01	1.0589377416236424E-4
remainder	0,994	0,994	3,59255262375E-5
roulette	0,865	-0,864	0,018374119078
roulette	0,908	-0,908	0,008394178608374
uniform	0,798	-0,798	0,040964266157
uniform	0,789	-0,789	0,0446002279365235
uniform	0,569	-0,569	0,18549685246
stochastic uniform	0,931	-0,931	0,004775469448751
stochastic uniform	0,996	0,996	1,77979245225E-5

2.2.3. Настройка масштабирования (Fitness Scaling)

Это функция, которая конвертирует достигаемые оптимизируемой функцией значения в значения, допустимые для оператора отбора.

—Rank (Ранг, Рейтинг) масштабирование – разработка рейтинга, то есть особям присваивается рейтинговый номер (для лучшей особи – единица, для следующей – двойка, и так далее).

— Proportional (Пропорциональное) масштабирование – задание вероятности пропорционально заданному числовому ряду для особей.

—Top (наибольшее) рейтинговое значение присваивается сразу нескольким наиболее выдающимся особям (их число указывается в виде параметра). Указанное количество особей – те, которые производят потомство.

— Shift linear (Линейный сдвиг) – масштабирует сырые оценки, так что ожидание наиболее приспособленной особи равно константе, которую можно указать как максимальную выживаемость, умноженную на средний балл. По результатам экспериментов самой эффективной для выбранной целевой функции оказалась функция Rank. То есть использование простого рейтинга без задания вероятности лучших особей и выбора ряда особей в качестве лучших работает эффективней. На рисунке 9 показан результат работы ГА при rank-функции масштабирования:

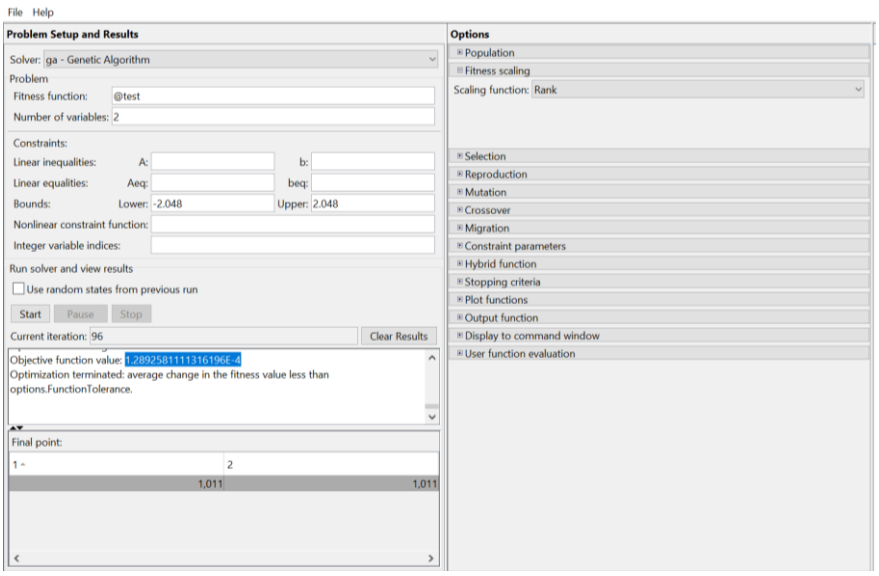


Рисунок 9 – Результат тестирования при работе rank-функции масштабирования

В таблице 7 представлены результаты работы ГА с разными функциями масштабирования.

Таблица 7 – Функции масштабирования

Функция масштабирования	x_1	x_2	Значение целевой функции
rank	0,909	-0,908	0,00837820111282
rank	1,011	1,011	1.2892581111316196E-4
proportional	0,858	0,857	0,020353185257354
proportional	1,271	-1,271	0,7366258735537
proportional	1,566	-1,566	0,320444483100246
top(0,2)	1,28	1,28	0,078479323836
top(0,4)	0,959	-0,959	0,001667141744
top(6)	0,974	-0,974	6,873869716E-4
shift linear(1)	0,625	0,624	0,14040963347787
shift linear(2)	1,279	1,279	0,0778842174911
shift linear(10)	0,761	0,76	0,057446098699675

2.2.4. Создание новых особей (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.

На рисунке 10 показан результат работы ГА при наличии 5% элитных особей и 80% кроссовере:

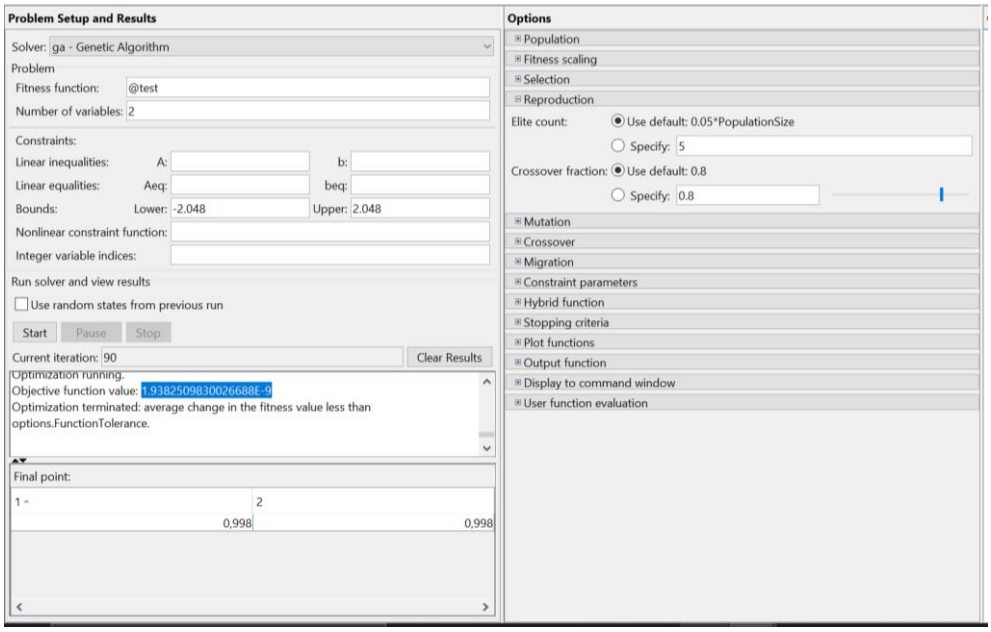


Рисунок 10 – результат тестирования при наличии 5% элитных особей и 80% кроссовере

Результаты тестирования представлены в таблице 8.

Таблица 8 – Создание новых особей

Элитные особи / Кроссовер	x_1	x_2	Значение целевой функции
5% / 80%	0,998	0,998	1.9382509830026688E-9
5% / 90%	0,932	0,932	0,004567896576
5% / 60%	0,993	-0,993	4,66985932256E-5
10% / 80%	0,985	-0,985	2,15003219809E-4
15% / 80%	-0,271	-0,31	1,668120267813801
30% / 40%	1,011	1,011	1,2728943653E-4
2% / 80%	1,049	1,049	0,002391995969451
2% / 60%	1,004	1,004	1,741906151373E-5
2% / 40%	1,019	1,019	3,48417844316E-4

2.2.5. Операторы 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.

Самым неэффективным методом оказался Gaussian с настроенными вручную параметрами, самым эффективным – Constraint dependent, метод по умолчанию. На рисунке 11 показан результат работы ГА при uniform(0,01) – операторе мутации:

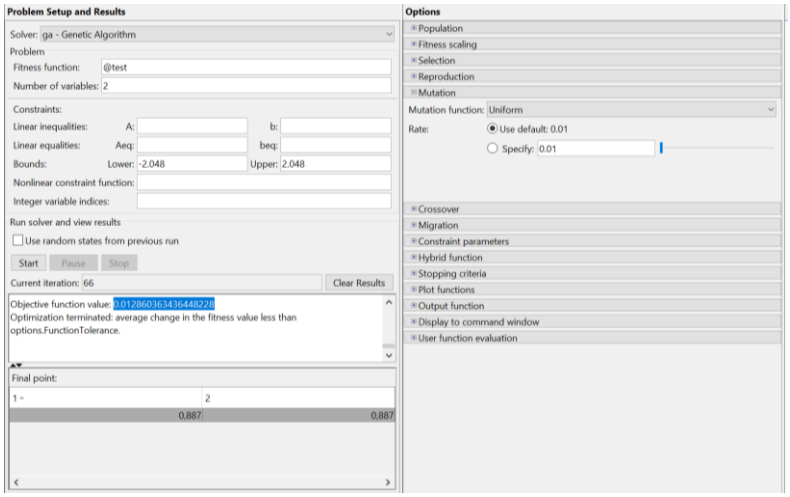


Рисунок 11- Результат тестирования при uniform(0,01)

Результаты экспериментов представлены в таблице 9.

Таблица 9 – Операторы мутации

Операторы мутации	x_1	x_2	Значение целевой функции
constraint dependent	1,044	1,044	0,001903785652
constraint dependent	0,996	0,996	1,5711167003862E-5
adaptive feasible	1,432	1,432	0,1866653875688348
adaptive feasible	0,989	-0,989	0,012860363436448228

uniform (0,01)	0,887	0,887	0,52035940710319
uniform(0,11)	0,745	0,744	0,065091949128479
uniform(0,71)	0,875	0,872	0,01971626422343
gaussian(1,1)	0,641	-0,637	0,1319363737920487
gaussian(0,1)	0,657	0,797	4,275341200287515
gaussian(1,0)	0,927	-0,929	0,00757777070136257
gaussian(0,0)	-0,079	0,006	1,1684962753379577

2.2.6. Оператор скрещивания (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.

На рисунке 12 показан результат работы ГА при scattered-операторе скрещивания:

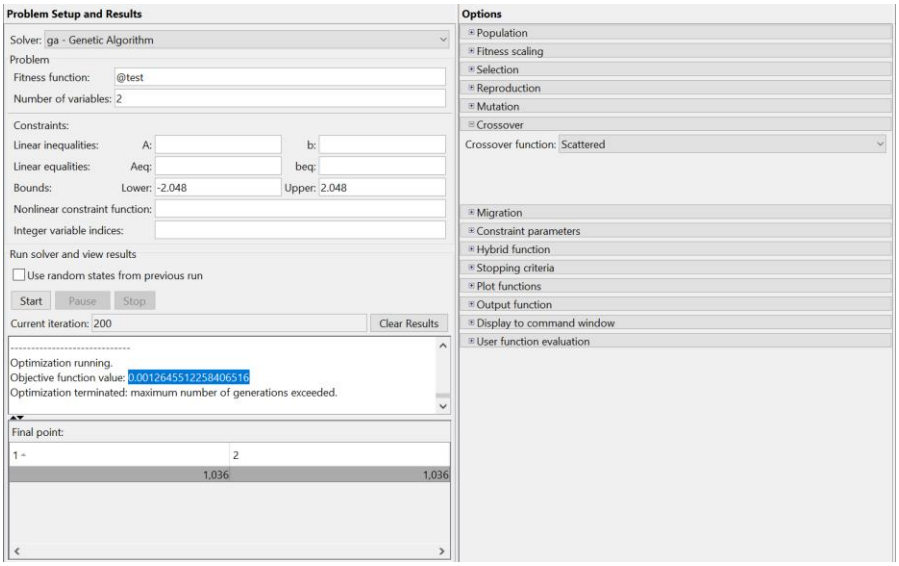


Рисунок 12 – Результат тестирования при scattered-операторе

Результаты тестирования представлены в таблице 10.

Таблица 10 – Операторы скрещивания

Оператор скрещивания	x_1	x_2	Значение целевой функции
constraint dependent	0,997	0,997	1,284382133628E-5
constraint dependent	0,936	-0,936	0,004077685293224807
scattered	0,955	-0,955	0,0019959509187223645
scattered	1,036	1,036	0,0012645512258406516
single point	1,001	-1,001	1,4664147415783E-6
single point	1,012	1,012	1,3673536898723E-4
two point	0,99	-0,99	9,8616146546599E-5
two point	1,128	1,129	0,01649908481921655
intermediate(1)	1,004	-1,004	1,79663804394644E-5
intermediate(0,7)	1	1	3,4915083582582665E-8
intermediate(0,3)	1	1	4,1079711633973E-13
heuristic(1,2)	1	1	9,296904162102E-12
heuristic(0,2)	0,944	0,945	0,003083945635667433
heuristic(1,8)	1	1	1,844198792856254E-9
Arithmetic	1	1	1,304839027902794E-9
Arithmetic	1	-1	3,4362070344493E-11

Выводы

По результатам тестирования работы генетического алгоритма с выбранной целевой функцией (формула (2)), были сделаны выводы.

$$f(x) = 100(x_1^2 - x_2^2)^2 + (1 - x_1)^2, \quad (2)$$

где -2,048 $\leq x_i \leq$2,048

1. Критерии остановки, которые можно настроить вручную, оптимальны в таком варианте, так как разброс значений особей небольшой (ограничение на интервале (-2,048; 2,048)).

2. При числе поколений больше 300-400, алгоритм работает существенно медленнее, однако разница в результатах не критична, так как в

большинстве случаев достаточно пригодное решение достигается алгоритмом в 200-300 поколении.

3. Лимит времени, оптимизации функции и количество мало отличающихся поколений для данной функции нецелесообразны, так как данная целевая функция быстро обрабатывается ГА, а также алгоритм быстро приближается к необходимому глобальному минимуму.

4. Тип популяции, начальная популяция, начальный рейтинг особей и начальный диапазон особей при изменении не показали разницы в результате работы ГА поэтому были оставлены без изменений.

5. Размер популяции – 200 особей. При значении, меньшем этого, ГА требуется больше поколений/времени, чтобы получить достаточно верные решения; при значении, большем этого, алгоритму в части случаев необходимо меньше поколений, чтобы получить решение, близкое к верному, в другой части случаев результат оказывается слишком далек от верного решения, так как алгоритм выдает ответы, которые являются локальными минимумами функции из-за слишком большого количества особей, так как особи обладающие даже недостаточной пригодностью переходят в следующее поколение и дают потомков (чья пригодность часто еще хуже чем у родителей).

6. Настройки масштабирования:

- Proportional, Top, Shift linear недостаточно эффективно масштабируют функцию, так как на результат влияет несколько особей, а в данном случае, при небольших разбросах значений не дает должной разницы в значениях пригодности, то есть и оператор отбора некорректно работает.
- Rank самый эффективный для данной функции, так как рейтинг дается каждой особи независимо от других, и даже особи с небольшими различиями в пригодности получают разные оценки и более эффективно отбираются оператором отбора, поэтому результат работы ГА с довольно большой вероятностью приближается к верному решению, а также простой выбор (без переопределения оценок для всей популяции, как в других случаях)

работает значительно быстрее, и на большом размере популяции эта тенденция прослеживается.

7. Оператор выбора:

- Tournament, Roulette, Stochastic uniform недостаточно эффективны, так как при работе с данной целевой функцией tournament не может провести полноценный отбор ввиду незначительных отличий оценок (если функция будет не гладкой этот метод будет более эффективен), roulette недостаточно эффективен, так как функция гладкая, вероятности появления значений недостаточно различаются, и алгоритм часто «пробегает мимо» глобального минимума, аналогично, как и при использовании stochastic uniform.
- Uniform недостаточно эффективен, так как в целом представляет неориентированный поиск, что не предоставляет абсолютно никакой гарантии на верное решение. c) Remainder более эффективен, так как используется два типа выбора – детерминированно выбирается целая часть, и уже для более пригодных результатов дробная часть выбирается методом рулетки, что в данном случае дает преимущество, так как при небольшом интервале он быстрее сужает значения особей к нужному глобальному минимуму и не пропускает его.

8. Воспроизводство особей :

Для данной функции были более эффективными настройки: 5% элитных особей, 80% участвовали в кроссовере, и, соответственно 15% участвовали в мутации. Если увеличивать процент элитных особей, то вероятность перехода недостаточно пригодных особей в следующее поколение слишком вырастает и ухудшает конечные результаты, так как в следующие поколения будут проходить даже особи с низкой пригодностью, но вошедшие в эту долю. И для данной функции кроссовер показал себя более эффективным способом улучшения особей, нежели мутация, так как функция имеет гладкий график, без локальных экстремумов. То есть при обмене генами в небольшом диапазоне значений особи быстрее эволюционируют, приближаясь к нужному

результату, нежели в результате случайных мутаций, которые могут только увести алгоритм от нужного результата. Если бы целевая функция имела локальные минимумы, то мутации были бы более востребованы для того, чтобы алгоритм смог выбираться из этих локальных минимумов.

Для выбранной целевой функции были определены оптимальные настройки (и функции, используемые при работе). А именно:

1) Критерии остановки

- Максимальное число поколений – 300;
- Остальные критерии стоит оставить без изменений, так как их перенастройка либо ухудшает результаты работы, либо не влияет на них;

2) Настройки популяции

- Размер популяции – 200 особей;
- С остальными настройками в этой вкладке стоит поступить так же, как и в предыдущем пункте;

3) Настройки масштабирования: функция масштабирования – Rank;

4) Оператор отбора – Remainder;

5) Создание новых особей – настройки по умолчанию;

6) Оператор мутации – настройки по умолчанию (либо тонкая настройка методов с параметрами);

7) Оператор скрещивания – Intermediate с параметром близким к 0;

8) Параметры в остальных вкладках стоит оставить без изменений в связи с теми же причинами.

Для наглядности работы ГА есть возможность построения графиков средой Matlab и в случае необходимости их так же можно включить в настройках алгоритма.

Список использованных источников

1. Вариационный генетический алгоритм решения NP-трудной задачи планирования / А.И. Дивеев, О.В. Бобр // Procedia Computer Science, 2017 - № 103 - с. стр. 52-58, XII Международный симпозиум Интеллектуальные системы 2016, INTELS 2016, 5-7 октября 2016, Москва, Россия.

2. Fluid Genetic Algorithm (FGA) / Ruholla Jafari-Marandi, Brian K. Smith // Journal of Computational Design and Engineering, 2017 - № 4 - с. 158-167.
3. Генетические алгоритмы в MATLAB [Электронный ресурс]: публикация от 07.01.2011 // «Хабрахабр» — крупнейший в Европе ресурс для ИТ-специалистов. – Режим доступа: https://habrahabr.ru/post/111417.
4. Вороновский Г.К., Махотило К.В., Петрашев С.Н., Сергеев С.А., Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности, Харьков, ОСНОВА, 1997. - 112с.
5. Раздел Genetic Algorithm and Direct Search Toolbox [Электронный ресурс]: Руководство пользователя Genetic Algorithm and Direct Search Toolbox // MATLAB.Exponenta - подробные авторские руководства по продуктам MathWorks. – Режим доступа: http://matlab.exponenta.ru.

References

1. Variational Genetic Algorithm for NP-hard Scheduling Problem Solution / A.I. Diveev, O.V. Bobr // Procedia Computer Science, 2017 - No. 103 - p. 52-58, XII International Symposium Intelligent Systems 2016, INTELS 2016, 5-7 October 2016, Moscow, Russia.
2. Fluid Genetic Algorithm (FGA) / Ruholla Jafari-Marandi, Brian K. Smith // Journal of Computational Design and Engineering, 2017 - No. 4 - p. 158-167.
3. Genetic algorithms in MATLAB [Electronic resource]: publication dated 01/07/2011 // "Habrahabr" - the largest resource in Europe for IT specialists. – Access mode: https://habrahabr.ru/post/111417.
4. Voronovsky G.K., Makhotilo K.V., Petrashev S.N., Sergeev S.A., Genetic algorithms, artificial neural networks and problems of virtual reality, Kharkov, OSNOVA, 1997. - 112p.
5. Section Genetic Algorithm and Direct Search Toolbox [Electronic resource]: User's Guide Genetic Algorithm and Direct Search Toolbox // MATLAB.Exponenta - detailed author's manuals on MathWorks products. – Access mode: http://matlab.exponenta.ru.

© Душина В.А. 2022 Научно-образовательный журнал для студентов и преподавателей «StudNet» №7/2022.

Для цитирования: Душина В.А. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ ЗАДАЧИ ОПТИМИЗАЦИИ МУЛЬТИМОДАЛЬНОЙ ФУНКЦИИ // Научно-образовательный журнал для студентов и преподавателей «StudNet» №7/2022.