МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7**

**по дисциплине Методы поисковой оптимизации**

Работу выполнила\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ромашкина А.А.

Факультет Компьютерных технологий и прикладной математики

Направление подготовки 02.03.02 Фундаментальная информатика и

информационные технологии курс 4

Краснодар

2022

СОДЕРЖАНИЕ

[Задание 3](#_Toc122436822)

[Алгоритм 3](#_Toc122436823)

[Результат работы программы 3](#_Toc122436824)

[Листинг 5](#_Toc122436825)

Тема работы: Бактериальный алгоритм.

# Задание

Разработать алгоритм бактериальной оптимизации обратной сферической функции.

Целевая функция имеет вид (так называемая гиперсфера):

.

Максимум такой функции находится в точке, когда все координаты равны 0.

# Алгоритм

Шаги алгоритма:

1. Генерация начальной популяции бактерий.
2. Выбирается бактерия с лучшим на данный момент значением функции.
3. Выполняется цикл по числу введенных итераций:
   1. Если новое найденное значение лучше старого лучшего, выполняется хемотаксис всех бактерий популяции. Находится бактерия с лучшим значением функции. Иначе генерируется случайное число. Если оно больше заданного параметра вероятности ликвидации, выполняется репродукция:
      1. Популяция бактерий сортируется по убыванию здоровья.
      2. Половина бактерий с меньшим значением здоровья удаляется.
      3. Другая половина дописывается к текущей популяции.
   2. Иначе ликвидация:
      1. Для заданного числа удаляемых бактерий случайным образом выбирается бактерия для удаления.
      2. Затем генерируется взамен новая случайная бактерия, которая добавляется в популяцию.
   3. Находится бактерия с лучшим значением функции.
   4. Происходит сравнение лучших найденных значений за всё время и новой найденной. Лучшее значение записывается в глобальную переменную.
4. Возврат лучшей найденной точки.

# Результат работы программы

Вся работа программы представлена на рисунках 1-2 ниже.

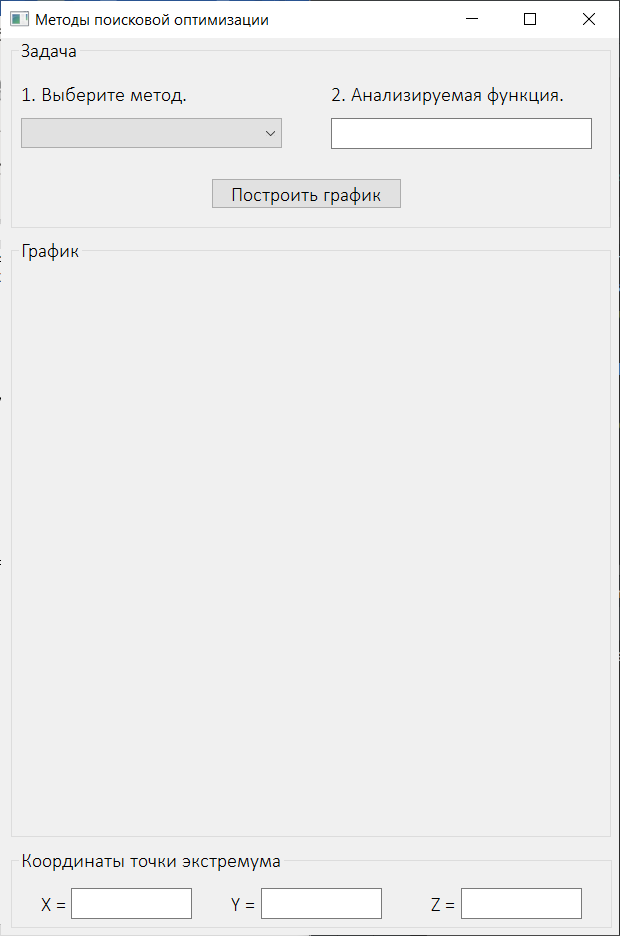


Рисунок 1 – Начальный вид программы при запуске

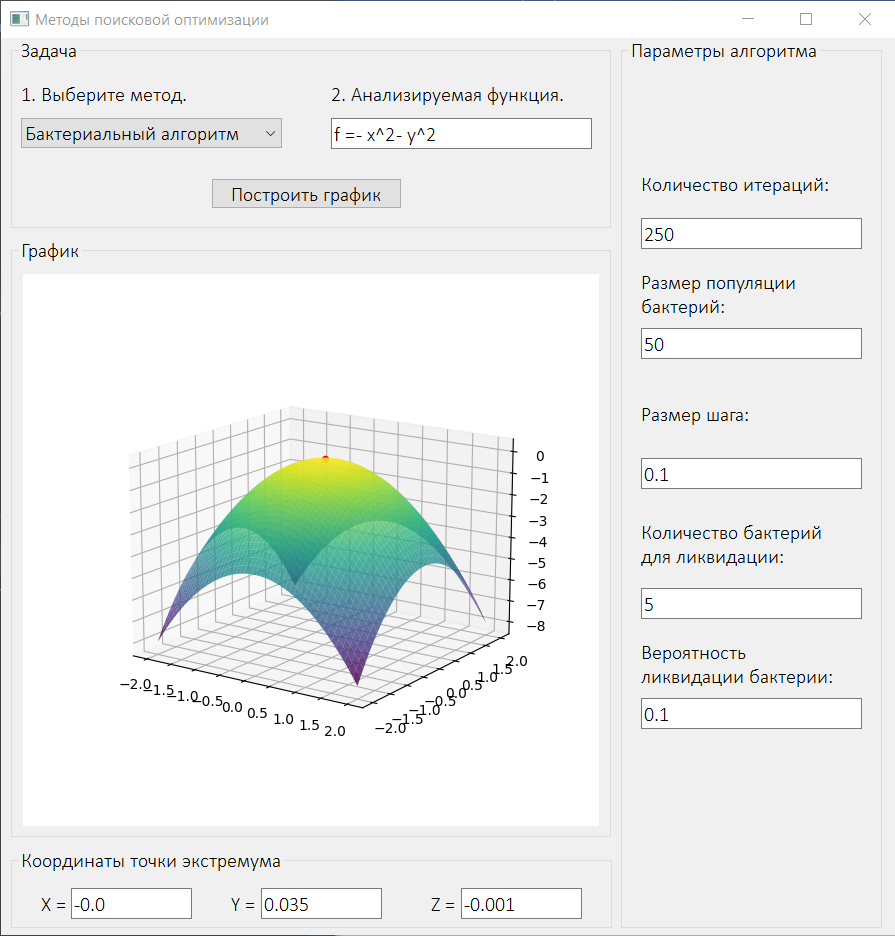


Рисунок 2 – Вид программы после выполнения выбранного алгоритма

# Листинг

import random

import math

def hipersphere(x, y):

return -(x \*\* 2 + y \*\* 2)

def mod(x):

return math.sqrt(x[0] \*\* 2 + x[1] \*\* 2)

def generate(num):

population = []

for i in range(num):

b = [[random.uniform(-3, 3), random.uniform(-3, 3)]]

b.append([random.uniform(-1, 1), random.uniform(-1, 1)])

b.append(hipersphere(b[0][0], b[0][1]))

population.append(b.copy())

return population

def chemotaxis(bacteria, lmbd):

operation = random.randint(0, 1)

if operation == 0:

# то движемся

bacteria[0][0] = bacteria[0][0] + lmbd \* bacteria[1][0] / (mod(bacteria[1]))

bacteria[0][0] = bacteria[0][1] + lmbd \* bacteria[1][1] / (mod(bacteria[1]))

else:

v = [random.uniform(-1, 1), random.uniform(-1, 1)]

bacteria[0][0] = bacteria[0][0] + lmbd \* v[0] / mod(v)

bacteria[0][1] = bacteria[0][1] + lmbd \* v[1] / mod(v)

bacteria[1] = v

bacteria[2] += hipersphere(bacteria[0][0], bacteria[0][1])

return bacteria

def reproduction(bacteries):

count = len(bacteries)

bacteries.sort(key=lambda a: a[2], reverse=True)

bacteries = bacteries[:int(count / 2)]

bacteries += bacteries.copy()

return bacteries

def elimination(bacteries, n):

for i in range(n):

x = random.randint(0, (len(bacteries) - 1))

del bacteries[x]

new\_bacteria = generate(1)

bacteries += (new\_bacteria.copy())

return bacteries

def get\_best\_solution(bacteries):

best\_solution = [bacteries[0][0].copy()]

best\_solution.append(hipersphere(best\_solution[0][0], best\_solution[0][1]))

for bac in bacteries:

if hipersphere(bac[0][0], bac[0][1]) > best\_solution[1]:

best\_solution[0] = bac[0].copy()

best\_solution[1] = hipersphere(bac[0][0], bac[0][1])

return best\_solution

def bacteria\_algorithm(num, iterations, lmbd, n, ep):

population = generate(num)

current\_best = get\_best\_solution(population)

global\_best = current\_best.copy()

bst = current\_best.copy()

for it in range(iterations):

# Пока ищется лучшее решение - делаем хемотаксис

if bst[1] <= current\_best[1]:

current\_best = bst.copy()

for i in range(len(population)):

population[i] = chemotaxis(population[i], lmbd)

bst = get\_best\_solution(population)

# Иначе репликация или ликвидация

else:

i = random.uniform(0, 1)

if i > ep:

population = reproduction(population)

else:

population = elimination(population, n)

bst = get\_best\_solution(population)

current\_best = bst.copy()

if global\_best[1] < current\_best[1]:

global\_best = current\_best.copy()

global\_best[1] = -global\_best[1]

# print(global\_best)

return global\_best[0]

# bacteria\_algorithm(10, 250, 0.1, 5, 0.1)