

Министерство образования Российской Федерации
МОСКВОСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Н.Э.БАУМАНА

Факультет: Информатика и системы управления (ИУ)
Кафедра: Информационная безопасность (ИУ8)

МЕТОДЫ ОПТИМИЗАЦИИ

Рубежный контроль №2 на тему:
«Исследование генетических алгоритмов в задачах поиска
экстремумов»

Вариант – 1

Преподаватель:
Коннова Н.С.

Студент:
Александров А. Н.

Группа:
ИУ8-34

Москва, 2020

Цель работы:

изучить основные принципы действия генетических алгоритмов на примере решения задач оптимизации функций двух переменных.

Постановка задачи:

Найти максимум функции $f(x, y)$ в области D с помощью простого (классического) генетического алгоритма.

За исходную популяцию принять 4 случайных точки. Хромосома каждой особи состоит из двух генов: значений координат x, y . В качестве потомков следует выбирать результат скрещивания лучшего решения со вторым и третьим в порядке убывания значений функции приспособленности с последующей случайной мутацией обоих генов.

В качестве критерия остановки эволюционного процесса задаться номером конечной популяции ($N \sim 10^1 \dots 10^2$). Визуализировать результаты расчетов.

Ход решения:

Исходная функция $f(x, y)$ имеет вид:

$$f(x, y) = \frac{\sin(x)}{1+x^2+y^2}$$

Область допустимых значений D :

$$D = (-2, 2) \times (-2, 2)$$

Для решения задачи была написана программа на языке **Python** (см. Приложение А):

1. *genetic_algorithm.py*

Содержит реализацию класса ***GeneticAlgorithm***, который инициализируется фит функцией и границами поиска экстремума.

Поля класса ***GeneticAlgorithm***:

Поле ***fit_func_***: инициализируется лямбда функцией от двух аргументов – *x* и *y*.

Поле ***iterations_count_***: содержит число итераций, которые совершает генетический алгоритм.

Поле ***bounds_***: содержит область допустимых значений аргументов *x* и *y*.

Поле ***species_count_***: содержит число особей в популяции.

Поле ***population_number_***: содержит номер текущей популяции.

Поле ***population_***: содержит текущую популяцию.

Поле ***max_result_***: содержит максимальное значение фит функции за популяцию.

Поле ***mean_result_***: содержит среднее значение фит функции за популяцию.

Методы класса ***GeneticAlgorithm***:

Метод ***PrintPopulation(self)***: выводит таблицу популяции, а также максимальное и среднее значения фит функции в ней.

Метод ***Solve(self)***: производит решение задачи указанное число итераций.

Метод ***InitPopulation(self)***: задание начальной популяции случайными значениями хромосом в пределах допустимых значений аргументов.

Метод ***Mutation(self)***: производит мутацию текущего поколения.

Метод ***CrossoverAndCalcFit(self)***: производит скрещивание особей промежуточной популяции путём кроссовера и вычисляет значение фит функции.

Метод ***VisualizePopulation(self)***: визуализирует поколение и сохраняет его в директорию /results.

График функции $f(x, y) = \sin(x) / (1 + x^2 + y^2)$

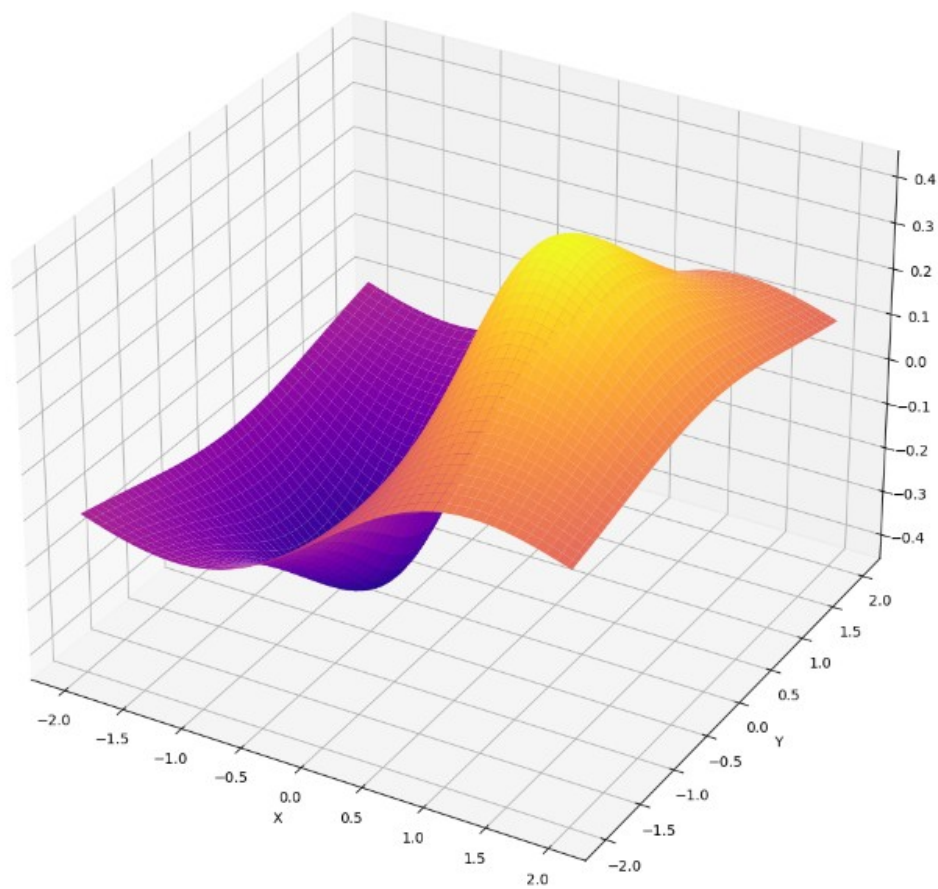


Рисунок 1 График исходной поверхности.

Начальное поколение:

Поколение: 0

| X | Y | FIT |
|-----------|-----------|-----------|
| 1.723671 | -1.07902 | 0.192458 |
| -1.894173 | -0.240393 | -0.204097 |
| 1.723671 | -0.638812 | 0.225693 |
| 0.348256 | -0.240393 | 0.28943 |

Максимальный результат: 0.28943
Средний результат: 0.125871

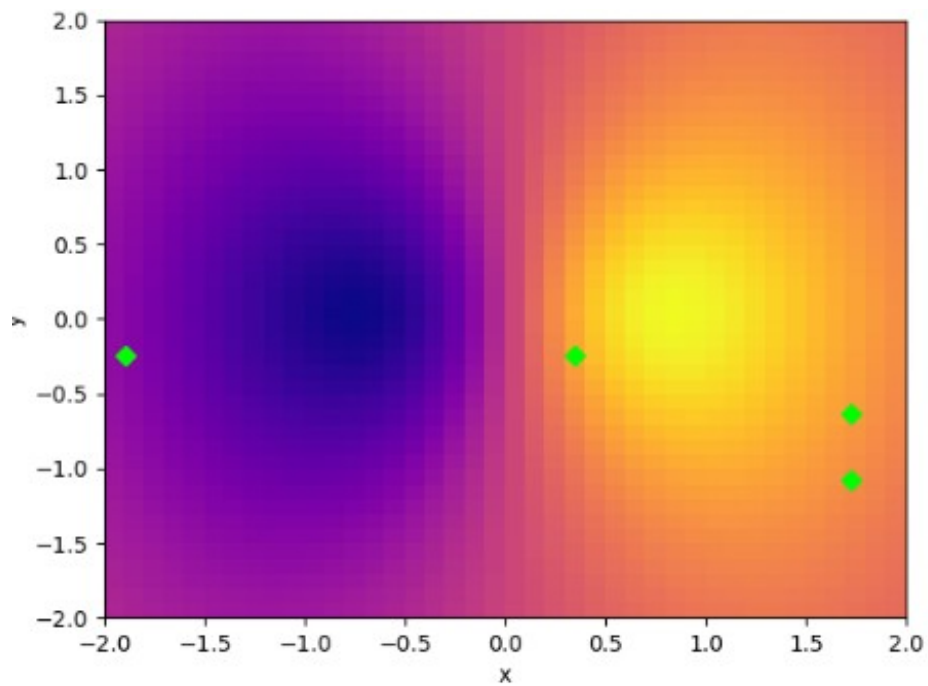


Рисунок 2 Начальное поколение

Первые 10 итераций алгоритма:

| | | | | | | |
|----------------------------------|----------|--|-----------|--|----------|--|
| Поколение: 1 | | | | | | |
| +-----+-----+-----+ | | | | | | |
| | X | | Y | | FIT | |
| +-----+-----+-----+ | | | | | | |
| | 0.348256 | | -1.07902 | | 0.14931 | |
| | 1.723671 | | -0.240393 | | 0.245316 | |
| | 0.348256 | | -0.638812 | | 0.223138 | |
| | 1.723671 | | -0.240393 | | 0.245316 | |
| +-----+-----+-----+ | | | | | | |
| Максимальный результат: 0.245316 | | | | | | |
| Средний результат: 0.21577 | | | | | | |
| Поколение: 2 | | | | | | |
| +-----+-----+-----+ | | | | | | |
| | X | | Y | | FIT | |
| +-----+-----+-----+ | | | | | | |
| | 1.723671 | | -0.638812 | | 0.225693 | |
| | 0.348256 | | -0.240393 | | 0.28943 | |
| | 1.723671 | | -0.240393 | | 0.245316 | |
| | 1.723671 | | -0.240393 | | 0.245316 | |
| +-----+-----+-----+ | | | | | | |
| Максимальный результат: 0.28943 | | | | | | |
| Средний результат: 0.251439 | | | | | | |
| Поколение: 3 | | | | | | |
| +-----+-----+-----+ | | | | | | |
| | X | | Y | | FIT | |
| +-----+-----+-----+ | | | | | | |
| | 0.348256 | | -0.240393 | | 0.28943 | |
| | 1.723671 | | -0.240393 | | 0.245316 | |
| | 0.348256 | | -0.240393 | | 0.28943 | |
| | 1.723671 | | -0.240393 | | 0.245316 | |
| +-----+-----+-----+ | | | | | | |
| Максимальный результат: 0.28943 | | | | | | |
| Средний результат: 0.267373 | | | | | | |
| Поколение: 4 | | | | | | |
| +-----+-----+-----+ | | | | | | |
| | X | | Y | | FIT | |
| +-----+-----+-----+ | | | | | | |
| | 1.662667 | | -0.147563 | | 0.263001 | |
| | 0.406113 | | -0.159661 | | 0.331851 | |
| | 1.662667 | | -0.296956 | | 0.258467 | |
| | 0.447112 | | -0.159661 | | 0.352834 | |
| +-----+-----+-----+ | | | | | | |
| Максимальный результат: 0.352834 | | | | | | |
| Средний результат: 0.301538 | | | | | | |
| Поколение: 5 | | | | | | |
| +-----+-----+-----+ | | | | | | |
| | X | | Y | | FIT | |
| +-----+-----+-----+ | | | | | | |
| | 0.466373 | | -0.190425 | | 0.358639 | |
| | 1.756958 | | -0.23524 | | 0.237244 | |
| | 0.466373 | | -0.168883 | | 0.360867 | |
| | 0.381655 | | -0.23524 | | 0.310123 | |
| +-----+-----+-----+ | | | | | | |
| Максимальный результат: 0.360867 | | | | | | |
| Средний результат: 0.316718 | | | | | | |

Рисунок 3 Первые 5 поколений.

| | | | |
|----------------------------------|-----------|----------|--|
| Поколение: 6 | | | |
| +-----+ | +-----+ | +-----+ | |
| X | Y | FIT | |
| +-----+ | +-----+ | +-----+ | |
| 0.466373 | -0.23524 | 0.353264 | |
| 0.381655 | -0.168883 | 0.317205 | |
| 0.466373 | -0.190425 | 0.358639 | |
| 0.466373 | -0.168883 | 0.360867 | |
| +-----+ | +-----+ | +-----+ | |
| Максимальный результат: 0.360867 | | | |
| Средний результат: 0.347494 | | | |
| Поколение: 7 | | | |
| +-----+ | +-----+ | +-----+ | |
| X | Y | FIT | |
| +-----+ | +-----+ | +-----+ | |
| 0.466373 | -0.23524 | 0.353264 | |
| 0.466373 | -0.168883 | 0.360867 | |
| 0.466373 | -0.190425 | 0.358639 | |
| 0.466373 | -0.168883 | 0.360867 | |
| +-----+ | +-----+ | +-----+ | |
| Максимальный результат: 0.360867 | | | |
| Средний результат: 0.35841 | | | |
| Поколение: 8 | | | |
| +-----+ | +-----+ | +-----+ | |
| X | Y | FIT | |
| +-----+ | +-----+ | +-----+ | |
| 0.466373 | -0.190425 | 0.358639 | |
| 0.466373 | -0.168883 | 0.360867 | |
| 0.466373 | -0.168883 | 0.360867 | |
| 0.466373 | -0.168883 | 0.360867 | |
| +-----+ | +-----+ | +-----+ | |
| Максимальный результат: 0.360867 | | | |
| Средний результат: 0.36031 | | | |
| Поколение: 9 | | | |
| +-----+ | +-----+ | +-----+ | |
| X | Y | FIT | |
| +-----+ | +-----+ | +-----+ | |
| 0.466373 | -0.168883 | 0.360867 | |
| 0.466373 | -0.168883 | 0.360867 | |
| 0.466373 | -0.168883 | 0.360867 | |
| 0.466373 | -0.168883 | 0.360867 | |
| +-----+ | +-----+ | +-----+ | |
| Максимальный результат: 0.360867 | | | |
| Средний результат: 0.360867 | | | |
| Поколение: 10 | | | |
| +-----+ | +-----+ | +-----+ | |
| X | Y | FIT | |
| +-----+ | +-----+ | +-----+ | |
| 0.466373 | -0.168883 | 0.360867 | |
| 0.466373 | -0.168883 | 0.360867 | |
| 0.466373 | -0.168883 | 0.360867 | |
| 0.466373 | -0.168883 | 0.360867 | |
| +-----+ | +-----+ | +-----+ | |
| Максимальный результат: 0.360867 | | | |
| Средний результат: 0.360867 | | | |

Рисунок 4 Поколения 5-10

Полученный результат 10 поколения: $x = 0.466, y = -0.169; f = 0.361$

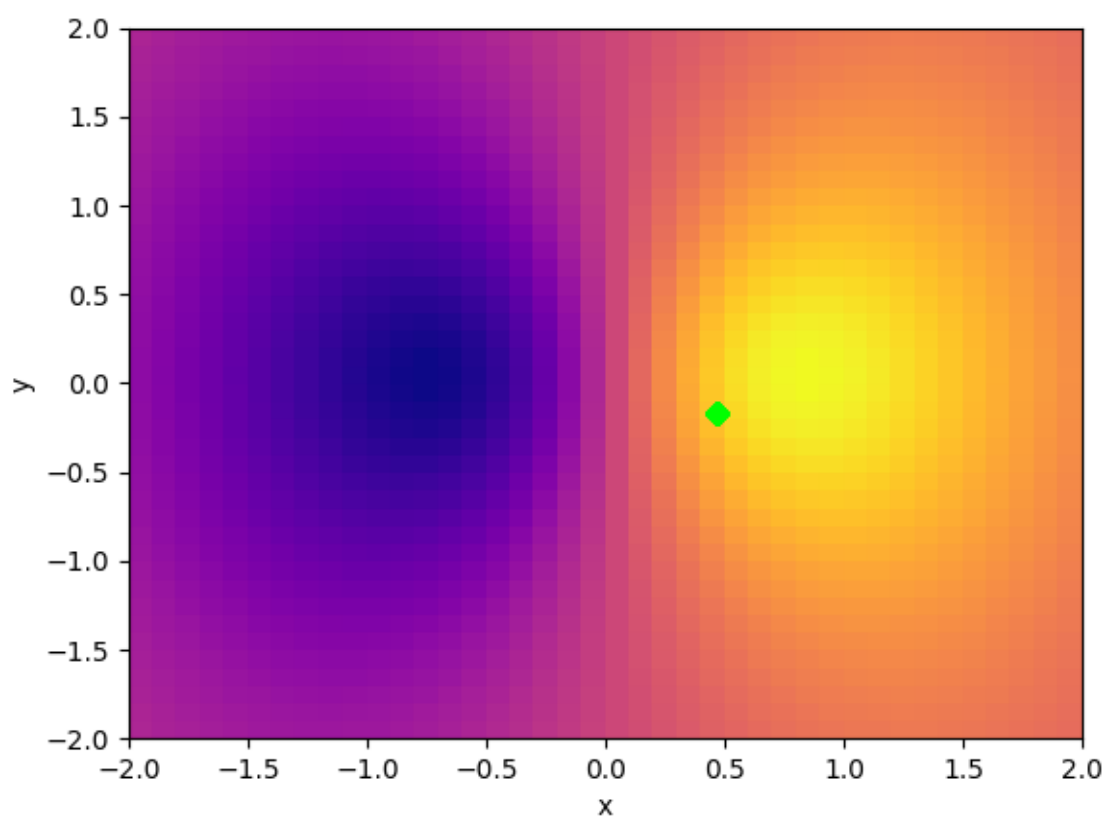


Рисунок 5 Визуализация поколения 10.

Популяции N = 10..100 с шагом 10:

```

Поколение: 10
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.466373 | -0.168883 | 0.360867 |
| 0.466373 | -0.168883 | 0.360867 |
| 0.466373 | -0.168883 | 0.360867 |
| 0.466373 | -0.168883 | 0.360867 |
+-----+-----+-----+
Максимальный результат: 0.360867
Средний результат: 0.360867

Поколение: 20
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.554139 | -0.185469 | 0.392265 |
| 0.554139 | -0.159542 | 0.394898 |
| 0.554139 | -0.159542 | 0.394898 |
| 0.554139 | -0.159542 | 0.394898 |
+-----+-----+-----+
Максимальный результат: 0.394898
Средний результат: 0.39424

Поколение: 30
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.588973 | -0.066582 | 0.411084 |
| 0.588286 | -0.066582 | 0.410907 |
| 0.588973 | -0.066582 | 0.411084 |
| 0.588973 | -0.066582 | 0.411084 |
+-----+-----+-----+
Максимальный результат: 0.411084
Средний результат: 0.41104

Поколение: 40
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.627848 | -0.185639 | 0.411159 |
| 0.502763 | -0.087963 | 0.382265 |
| 0.627848 | -0.082042 | 0.419298 |
| 0.563936 | -0.087963 | 0.403177 |
+-----+-----+-----+
Максимальный результат: 0.419298
Средний результат: 0.403975

Поколение: 50
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.764859 | -0.16668 | 0.42934 |
| 0.706055 | -0.016062 | 0.432913 |
| 0.764859 | -0.124542 | 0.432631 |
| 0.706055 | -0.016062 | 0.432913 |
+-----+-----+-----+
Максимальный результат: 0.432913
Средний результат: 0.431949

```

Рисунок 6 Поколения 10, 20, 30, 40, 50.

```

Поколение: 60
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.821945 | 0.05469 | 0.436362 |
| 0.745884 | -0.113321 | 0.432468 |
| 0.821945 | 0.031595 | 0.436881 |
| 0.673733 | -0.113321 | 0.425365 |
+-----+-----+-----+
Максимальный результат: 0.436881
Средний результат: 0.432769

Поколение: 70
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.797664 | -0.109481 | 0.434233 |
| 0.850419 | 0.053489 | 0.435414 |
| 0.797664 | 0.053489 | 0.436651 |
| 0.785819 | 0.053489 | 0.436569 |
+-----+-----+-----+
Максимальный результат: 0.436651
Средний результат: 0.435717

Поколение: 80
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.797664 | 0.053489 | 0.436651 |
| 0.797664 | 0.053489 | 0.436651 |
| 0.797664 | 0.053489 | 0.436651 |
| 0.797664 | 0.053489 | 0.436651 |
+-----+-----+-----+
Максимальный результат: 0.436651
Средний результат: 0.436651

Поколение: 90
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.766477 | 0.154315 | 0.430461 |
| 0.917409 | -0.037012 | 0.430833 |
| 0.766477 | -0.037012 | 0.436541 |
| 0.886586 | -0.037012 | 0.433544 |
+-----+-----+-----+
Максимальный результат: 0.436541
Средний результат: 0.432845

Поколение: 100
+-----+-----+-----+
|   X   |   Y   |   FIT   |
+-----+-----+-----+
| 0.766477 | -0.037012 | 0.436541 |
| 0.766477 | -0.037012 | 0.436541 |
| 0.766477 | -0.037012 | 0.436541 |
| 0.766477 | -0.037012 | 0.436541 |
+-----+-----+-----+
Максимальный результат: 0.436541
Средний результат: 0.436541

```

Рисунок 7 Поколения 60, 70, 80, 90, 100

Полученный результат 100 поколения: $x = 0.766, y = -0.037; f = 0.437$

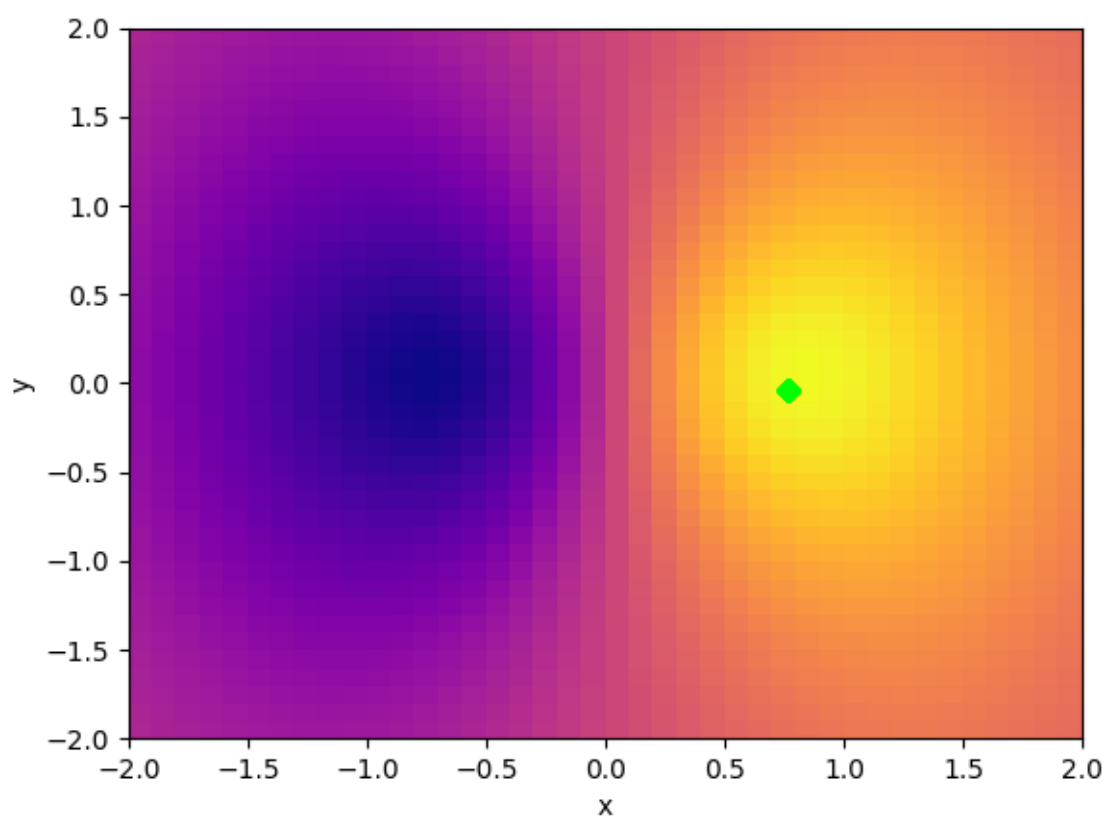


Рисунок 8 Поколение 100

Вывод:

Проделав работу, я изучил основные принципы действия генетических алгоритмов на примере поиска экстремума функции двух переменных; изучил основные шаги алгоритма, возможные критерии останова, виды алгоритмов селекции, операторов кроссовера и мутации.

По результатам численного эксперимента мы видим, что на 10 поколении результат ещё далековат от необходимого экстремума, но на 100 - совсем близок к идеальному.

Идея эволюционных алгоритмов в том, чтобы найти некое приближенное к лучшему, к оптимальному решению, которое скорее всего будет нас удовлетворять. Им можно найти применение для решения различных инженерных и практических задач.

К недостаткам генетических алгоритмов можно отнести ограниченность в способности нахождения верного решения. Они не могут регулировать логику развития поиска, существует вероятность, что в некоторых случаях алгоритм будет существенно ошибаться.

Приложение А

Код программы

Файл “main.py”

```
# Copyright 2020 Alexey Alexandrov <sks2311211@yandex.ru>

import genetic_algorithm as genetic
import numpy as np

# Вариант 1.
#  $f(x, y) = \sin(x) / (1 + x^2 + y^2)$ 

if __name__ == '__main__':
    fit_function = lambda x, y: np.sin(x) / (1 + x ** 2 + y ** 2)
    bounds = [-2., 2., -2., 2.]
    genetic.GeneticAlgorithm(fit_function, bounds).Solve()
```

Файл “genetic_algorithm.py”

```
# Copyright 2020 Alexey Alexandrov <sks2311211@yandex.ru>
import os
import random

import imageio
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import matplotlib.pyplot as plt

import numpy as np
from prettytable import PrettyTable

ITERATIONS_COUNT = 101
SPECIES_COUNT = 4
MUTATE_CHANCE = 0.25
MUTATE_CONSTANT = 5
ROUND_CONSTANT = 6

class GeneticAlgorithm:
    def __init__(self, fit_func, bounds):
        """Инициализация задачи по фит функции и границам
        диапазона."""
        self.fit_func_ = fit_func
        self.iterations_count_ = ITERATIONS_COUNT
```

```

self.bounds_ = bounds
self.species_count_ = SPECIES_COUNT
self.population_number_ = 0
self.population_ = self.InitPopulation()

self.max_result_ = np.max(self.population[:, 2])
self.mean_result_ = np.mean(self.population[:, 2])

def PrintPopulation(self):
    """Вывод таблицы."""
    print(f"Поколение: {self.population_number_}")
    table = PrettyTable()
    table.field_names = ["X", "Y", "FIT"]

    for i in range(self.species_count_):
        table.add_row(list(np.round(self.population[i], ROUND_CONSTANT)))

    self.population_number_ += 1
    print(table)

    print(f"Максимальный результат: {round(self.max_result_,
ROUND_CONSTANT)}")
    print(f"Средний результат: {round(self.mean_result_,
ROUND_CONSTANT)}")

def Solve(self):
    """Решение задачи."""
    for i in range(self.iterations_count_):
        self.VisualizePopulation()
        self.CrossoverAndCalcFit()
        self.PrintPopulation()

        # Mutate in 25% of population.
        if random.uniform(0, 1) <= 0.25:
            self.Mutation()

    self.CreateGifVisualization()

def InitPopulation(self):
    """Начальная популяция."""
    population = np.random.rand(self.species_count_, 3)

    # Столбец X.
    population[:, 0] = self.bounds_[0] + population[:, 0] * (self.bounds_[1] -
self.bounds_[0])
    # Столбец Y.

```

```

    population[:, 1] = self.bounds_[2] + population[:, 1] * (self.bounds_[3] -
self.bounds_[2])
    # Столбец FIT.
    population[:, 2] = self.fit_func_(population[:, 0], population[:, 1])

    return population

def Mutation(self):
    """Мутация"""
    delta = (np.random.rand(*self.population_.shape) - 0.5) /
MUTATE_CONSTANT
    self.population_ = self.population_ + delta
    x_column = self.population_[0]
    y_column = self.population_[1]

    satisfying_x = np.where(np.logical_and(x_column > self.bounds_[0],
x_column < self.bounds_[1]))
    satisfying_y = np.where(np.logical_and(y_column > self.bounds_[2],
y_column < self.bounds_[0]))
    if len(satisfying_x) != len(x_column) or len(satisfying_y) !=
len(y_column):
        self.population_ = self.population_ - delta

def CrossoverAndCalcFit(self):
    """Кроссовер и вычисление значений фит функции."""
    curr_population = self.population_[self.population[:,
2].argsort()].copy()

    x3 = curr_population[3][0]
    y3 = curr_population[3][1]
    x2 = curr_population[2][0]
    y2 = curr_population[2][1]
    x1 = curr_population[1][0]
    y1 = curr_population[1][1]

    new_population = np.array(
        [[x3, y1, self.fit_func_(x3, y1)], [x1, y3, self.fit_func_(x1, y3)],
        [x3, y2, self.fit_func_(x3, y2)], [x2, y3, self.fit_func_(x2, y3)]]

    self.max_result_ = np.max(new_population[:, 2])
    self.mean_result_ = np.mean(new_population[:, 2])

    self.population_ = new_population.copy()

def VisualizePopulation(self):
    if not os.path.exists('results/'):

```

```

    os.makedirs('results/')
    x0, x1 = self.bounds_[:2]
    x = np.arange(x0, x1 + 0.1, 0.1)
    y0, y1 = self.bounds_[2:]
    y = np.arange(y0, y1 + 0.1, 0.1)
    x, y = np.meshgrid(x, y)
    z = self.fit_func_(x, y)

    x_p, y_p = np.transpose(self.population_[ :, :2])
    plt.title(f"Поколение {self.population_number_}")
    plt.xlabel('x')
    plt.ylabel('y')
    plt.pcolormesh(x, y, z, cmap=cm.plasma)
    plt.plot(x_p, y_p, marker='D', linestyle="", color='lime')
    plt.savefig('results/' + str(self.population_number_) + '.png')
    # plt.show()
    plt.clf()

```