МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

|  |
| --- |
| КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ |

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| д-р техн. наук, профессор |  |  |  | Ю.А. Скобцов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| **Оценка сложности программных проектов** |
| по дисциплине: Эволюционные методы проектирования программно-информационных систем |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | Z1431 |  |  |  | М.Д.Быстров |
|  | номер группы |  | подпись, дата |  | инициалы, фамилия |
| Студенческий билет № | 2021/3572 | |  |  |  |

Санкт-Петербург 2025

Оглавление

[Индивидуальное задание 3](#_Toc196942673)

[Краткие теоретические сведения 4](#_Toc196942674)

[Результаты работы программы 7](#_Toc196942675)

[Ответ на контрольный вопрос 14](#_Toc196942676)

[Вывод 15](#_Toc196942677)

[Список литературы 16](#_Toc196942678)

[Приложение 1 Листинг программы 17](#_Toc196942679)

# Индивидуальное задание

1. Разобраться в теоретическом описании математического метода оценки стоимости программного проекта – модели СОСОМО.

2. Из приведенной выше табл.8.1 (или табл.8.2) экспериментальных данных (программных проектов НАСА) отобрать из 18 проектов в качестве обучающего множества 13 (40) проектов.

3. В соответствии с вариантом лабораторной работы определить тип используемого эволюционного алгоритма (генетический или роевой алгоритм, генетическое программирование), кодирование потенциального решения, вид ошибки в целевой функции, вид генетических операторов кроссовера, мутации и репродукции.

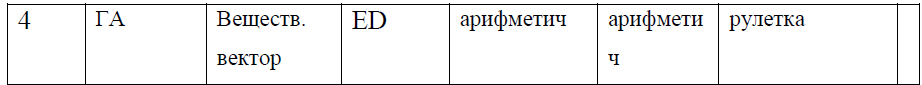
4. Отработать алгоритм решения задачи с помощью заданного метода на обучающем множестве.

5. Разработать программу на любом доступном вам языке программирования, включающую в себя реализацию пользовательского интерфейса в виде диалогового меню, реализацию алгоритма решения поставленной задачи заданным методом.

6.Протестировать разработанную программу: вычислить заданный тип ошибки на тестовом множестве – оставшихся 5 (из 18) проектов табл.8.1 (или табл.8.2).

5.Выполнить вывод полученного решения в виде текста и графиков.

Вариант 4



# Краткие теоретические сведения

Одной из самых популярных моделей, используемых для оценки сложности проектируемого программного обеспечения (ПО) , является модель COCOMO (COnstructive COst Model), предложенная Boehm [1,2]. Эта модель разработана на основе фактически статистики 63 проектов ПО (НАСА). Модель позволяет определить математическую зависимость между сложностью ПО, выраженную в килостроках кода, и затратами на его разработку, которые оцениваются в человеко-месяцах.

Ядром модели является следующая формула Ef=aLb , где L - длина кода ПО в килостроках; Ef – оценка сложности проекта в человеко-месяцах; a и b – коэффициенты (параметры) модели, которые для различных типов ПО имеют различные значения. Основная проблема модель COCOMO заключается в том, что она не обеспечивает реальных оценок на затраты при проектировании ПО в современных условиях. Т.е. оценка программного обеспечения на основе существующих параметров не всегда дает точный результат; из-за этого часто требуется настройка параметров для получения более точных результатов.

Поэтому в настоящее время идет активный поиск новых моделей (или развития и модификаций существующих). Это ограничение модели COCOMO можно преодолеть путем применения методов искусственного интеллекта, таких как искусственные нейронные сети, генетические алгоритмы и другие метаэвристики.

В данной лабораторной работе для определения значений коэффициентов a и b используются генетический или роевой алгоритм в соответствии с заданным вариантом. Фактически задача сводится к машинному обучению на заданной обучающей выборке. В этом случае обучающая выборка строится на основе следующей таблицы, которая дает реальные данные для 18 проектов НАСА, на основе которых мы ищем зависимость между L и Ef.

Напомним, что для того, чтобы применить генетический алгоритм для решения некоторой проблемы необходимо, прежде всего, определить:

1. Кодирование (представление потенциального решения);

2. Для определенного кодирования выбрать или разработать генетические операторы кроссовера, мутации и репродукции.

3. Фитнесс-функцию из условия задачи.

4. Определить параметры ГА: число особей в популяции, значения вероятностей кроссовера Pc и Pm .

В данном случае потенциальное решение представляется вектором значений параметров (a,b). Значения каждого параметра лежат в некотором диапазоне.

Для кодирования значений вектора (a,b) можно использовать как двоичное кодирование, так и непосредственное представление потенциального решения в виде вектора вещественных чисел (a,b). Кодирование решения определяется вариантом курсовой работы согласно приведенной далее таблице. В случае двоичного кодирования можно использовать стандартный 1-точечный, 2-точечный (или однородный) кроссовер и стандартный оператор мутации. В случае вещественного кодирования следует использовать какой-либо вещественный кроссовер (например, в виде линейной комбинации родительских векторов) и вещественную мутацию. Значения параметров ГА следует подобрать экспериментально в ходе эксперимента.

**Фитнесс-функция**

В качестве фитнесс-функции в данном случае следует взять различие

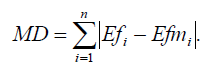
между реальными значениями стоимостей проектов и модельными

значениями (оценками) стоимостей этих же проектов, которые вычислены

согласно приведенной формуле с найденными с помощью ГА

коэффициентами a и b. Это различие (расстояние между оценками) можно

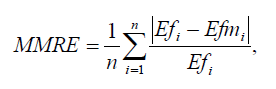
оценить по-разному - в различной метрике. Можно взять, например, метрику абсолютных значений (Манхэттен – метрика городских кварталов), где это различие определяется с помощью следующей формулы



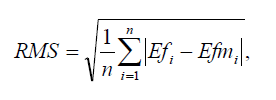
Здесь Efi – реальная (измеренная) стоимость i-го проекта в человеко-

месяцах и Efmi – модельная оценка того же проекта, вычисленная с помощью приведенной формулы с найденными путем применения ГА коэффициентами a и b. Для оценки различия можно использовать и другие метрики [18], например:

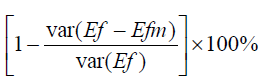
* среднее значение относительной погрешности (MMRE)



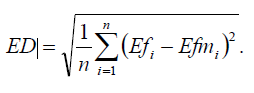
* корень квадратный среднеквадратичной ошибки (RMS)



- отклонение (дисперсия) (VAF):

,

- Евклидово расстояние:

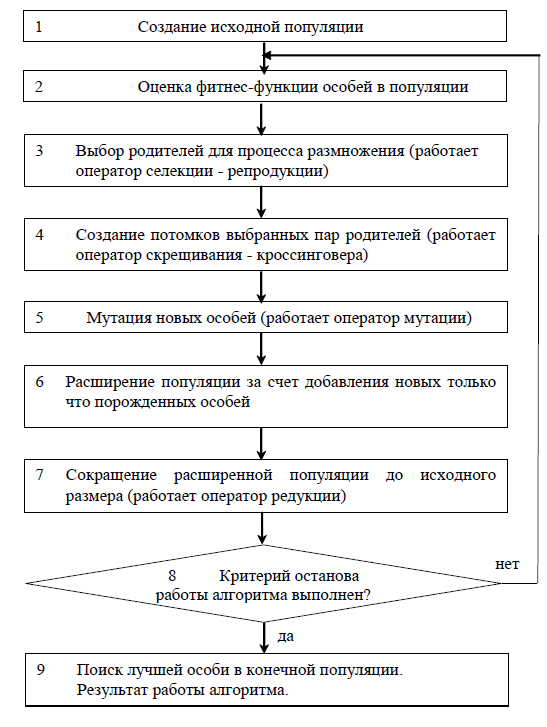


# Результаты работы программы

Таблица 1 Входные данные

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | L – в  килостроках | Ef -реальная стоимость в человекомесяцах | COCOMO  с ошибкой  MRE | ИНС  с MRE | ГА  с MRE | Гибрид с MRE |
| 1 | 2.2 | 8.4 | 24.15 | 13.65 | 8.95 | 6.32 |
| 2 | 3.5 | 10.8 | 3.95 | 5.26 | 4.69 | 1.13 |
| 3 | 5.5 | 18 | 7.36 | 5.21 | 6.75 | 4.35 |
| 4 | 6 | 24 | 58.88 | 34.10 | 27.63 | 28.02 |
| 5 | 9.7 | 25.2 | 20.05 | 11.50 | 13.49 | 7.61 |
| 6 | 7.7 | 31.2 | 23.91 | 12.35 | 7.54 | 12.42 |
| 7 | 11/3 | 36 | 30.83 | 17.45 | 12.45 | 13.35 |
| 8 | 8.2 | 36 | 29.55 | 16.68 | 14.23 | 11.21 |
| 9 | 6.5 | 42 | 28.32 | 18.52 | 11.64 | 13.42 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 | 8 | 42 | 22.22 | 13.21 | 15.47 | 9.34 |
| 11 | 20 | 48 | 27.21 | 14.65 | 16.32 | 12.16 |
| 12 | 10 | 48 | 41.66 | 23.98 | 19.84 | 19.84 |
| 13 | 15 | 48 | 46.19 | 28.04 | 23.11 | 26.74 |
| 14 | 10.4 | 50 | 34.90 | 25.47 | 17.02 | 21.95 |
| 15 | 13 | 60 | 9.36 | 6.53 | 5.31 | 7.15 |
| 16 | 14 | 60 | 25.88 | 15.41 | 17.54 | 8.46 |
| 17 | 19.7 | 60 | 6,10 | 7.21 | 4.21 | 2.54 |
| 18 | 32.5 | 60 | 93.91 | 47.35 | 56.47 | 36.10 |
| 19 | 31.5 | 60 | 3.81 | 6.52 | 5.46 | 1.07 |
| 20 | 12.5 | 62 | 27.96 | 13.11 | 10.84 | 4.31 |
| 21 | 15.4 | 70 | 22.51 | 10.13 | 12.76 | 7.02 |
| 22 | 20 | 72 | 60.76 | 45.68 | 33.82 | 27.11 |
| 23 | 7.5 | 72 | 41.75 | 32.61 | 24.15 | 15.04 |
| 24 | 16.3 | 82 | 29.79 | 23.40 | 17.37 | 7.46 |
| 25 | 15 | 90 | 39.54 | 27.68 | 21.51 | 19.01 |
| 26 | 11.4 | 98.8 | 42.04 | 25.10 | 19.07 | 21.74 |
| 27 | 21 | 107 | 36.75 | 24.55 | 16.53 | 9.02 |
| 28 | 16 | 114 | 34.48 | 24.55 | 16.53 | 9.92 |
| 29 | 25.9 | 117.6 | 27.85 | 19.36 | 11.57 | 17.09 |
| 30 | 24.6 | 117.6 | 31.65 | 21.87 | 16.34 | 14.82 |
| 31 | 29.5 | 120 | 18.94 | 11.15 | 7.13 | 6.44 |
| 32 | 19.3 | 155 | 35.78 | 17.30 | 21.06 | 16.72 |
| 33 | 32.6 | 170 | 29.88 | 19.54 | 15.19 | 5.68 |
| 34 | 35.5 | 192 | 32.10 | 16.35 | 8.37 | 13.06 |
| 35 | 38 | 210 | 28.46 | 13.19 | 19.50 | 15.43 |
| 36 | 48.5 | 239 | 24.31 | 8.43 | 12.07 | 7.94 |
| 37 | 47.5 | 252 | 37.81 | 21.36 | 18.64 | 11.83 |
| 38 | 70 | 278 | 21.28 | 9.42 | 11.46 | 6.24 |
| 39 | 66.6 | 300 | 23.76 | 11.30 | 16.79 | 9.22 |
| 40 | 66.6 | 352.8 | 35.17 | 19.25 | 11.20 | 13.62 |
| 41 | 50 | 370 | 36.90 | 23.54 | 13.48 | 7.42 |
| 42 | 79 | 400 | 45.74 | 31.29 | 22.97 | 18.06 |
| 43 | 90 | 450 | 38.29 | 20.11 | 31.73 | 15.94 |
| 44 | 78 | 571.4 | 24.50 | 13.64 | 8.03 | 5.21 |
| 45 | 100 | 215 | 120.66 | 86.14 | 61.42 | 51.04 |
| 46 | 150 | 324 | 49.50 | 26.80 | 13.09 | 23.83 |
| 47 | 100 | 360 | 44.97 | 17.67 | 25.07 | 12.62 |
| 48 | 100 | 360 | 15.85 | 6.23 | 8.62 | 9.84 |
| 49 | 190 | 420 | 1.89 | 4.87 | 3.84 | 2.65 |
| 50 | 115.8 | 480 | 11.37 | 16.49 | 5.32 | 5.42 |
| 51 | 101 | 750 | 19.87 | 10.67 | 6.46 | 12.71 |
| 52 | 161.1 | 815 | 4.76 | 10.25 | 8.41 | 5.95 |
| 53 | 284.7 | 973 | 38.36 | 21.43 | 17.09 | 10.14 |
| 54 | 227 | 1181 | 3.93 | 2.36 | 6.31 | 4.62 |
| 55 | 177.9 | 1228 | 3.64 | 9.84 | 5.08 | 2.06 |
| 56 | 282.1 | 1368 | 17.21 | 9.46 | 11.36 | 7.92 |
| 57 | 219 | 2120 | 29.00 | 21.03 | 15.81 | 8.31 |
| 58 | 423 | 2300 | 25.78 | 16.07 | 7.44 | 9.02 |
| 59 | 302 | 2400 | 0.46 | 3.24 | 5.64 | 2.54 |
| 60 | 370 | 3240 | 25.21 | 8.62 | 3.21 | 6.87 |



*Рисунок 1 Блок-схема генетического алгоритма*

Для кодирования хромосом особей используется представление в виде вектора вещественных чисел.

В качестве оператора репродукции используется оператор «рулетка» .

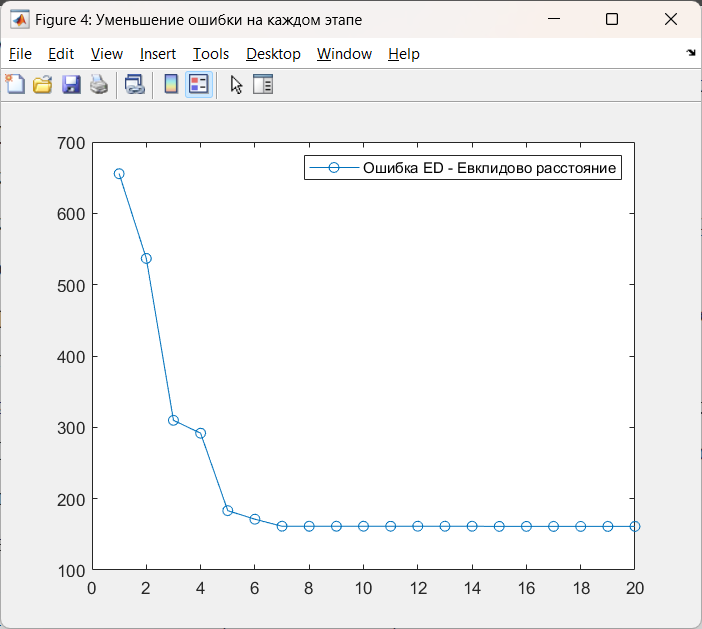
Кроссовер – смешанный кроссовер, каждый из элементов вектора потомка принимает случайное значение из диапазона [cmin–I\*alpha, cmax+I\*alpha], где Cmin – минимальная из хромосом родителя, Cmax – максимальная из хромосом, I – расстояние между хромосомами, alpha = 0,25.

Оператор мутации – случайная мутация, когда элемент вектора принимает случайное значение в интервале [C – delta; C + delta], где C – значение хромосомы, delta = 1.

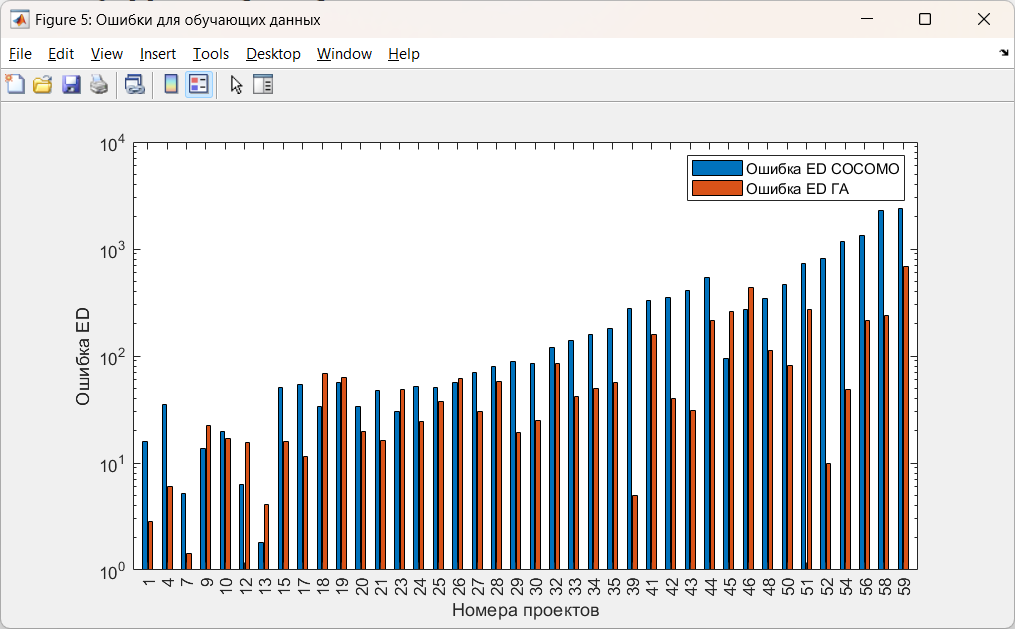
Отбор обучающего и тестового множества происходит случайным образом при каждом запуске программы. Выбирается 40 случайных номеров из диапазона [1; 60], проекты под этими номерами становятся обучающими, остальные проекты становятся тестовыми.

Используемый тип ошибки – Евклидово расстояние.

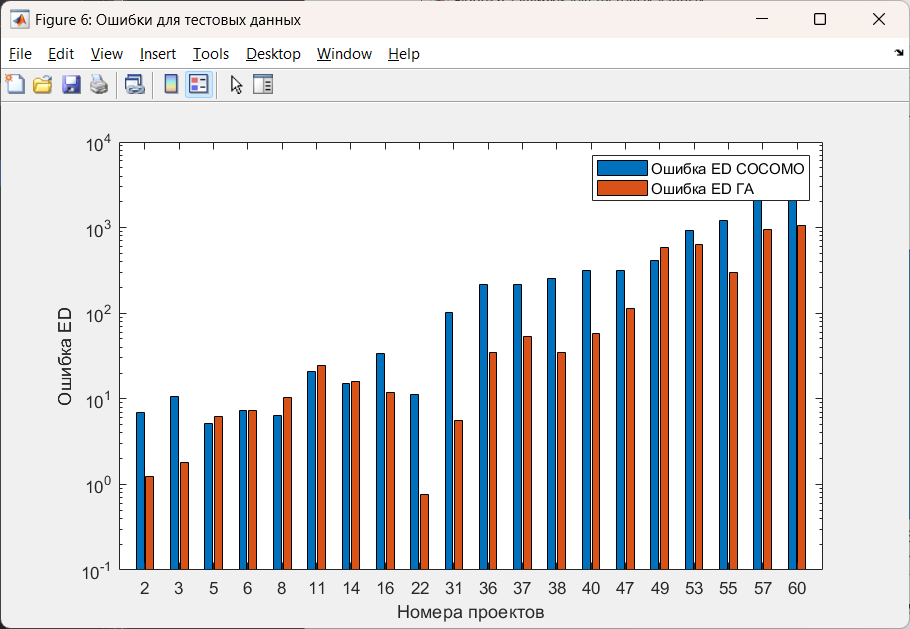
Параметры ГА: размер популяции – 100 особей, вероятность кроссовера – 0.5, вероятность мутации – 0.1. Количество эпох – 20.



*Рисунок 2 График обучения*



*Рисунок 3 Значения ошибки на обучающем множестве проектов*



*Рисунок 4 Значения ошибки на тестовом множестве.*

Найденные значения коэффициентов: a = 2.227, b = 1.164.

Формула Ef:

# Ответ на контрольный вопрос

4. Какие данные можно использовать при обучении (поиске

коэффициентов a, b)?

Задача поиска коэффициентов – максимально приблизиться к реальному значению трудозатрат для набора проектов.

Функция расчета трудозатрат зависит от метрики сложности проекта (измеряемой в килостроках).

Таким образом, для обучения (поиска коэффициентов a и b) могут быть использованы данные о сложности и фактических трудозатратах по проектам, входящим в обучающий набор данных.

# Вывод

В ходе выполнения четвертой лабораторной работы была написана программа в среде MATLAB для поиска оптимальных коэффициентов модели COCOMO для оценки сложности программных проектов.

Используется генетический алгоритм с вещественным кодированием, оператором репродукции «рулетка», смешанным оператором скрещивания и случайной мутацией. Также применена стратегия «элитизма».

Результаты, полученные при запуске написанной программы, позволяют сделать вывод о возможности улучшения модели COCOMO с помощью обучения на наборе данных. И на обучающих, и на тестовых данных вычисленные коэффициенты позволили получить меньший показатель ошибки (Евклидово расстояние) для большинства проектов.

Сформированы графики, показывающие ошибки в ходе поиска коэффициентов и разницу между ошибками COCOMO и ГА.

Приведена функция поиска трудозатрат с итоговыми коэффициентами.

# Список литературы

Литература

1. Ю.А.Скобцов. Эволюционные методы в программной инженерии.Учебное пособие.СПб:ГУАП,2020.-130с.

2. Ю.А.Скобцов. Генетические алгоритмы в программной инженерии.Москва Вологда:Инфра-Инженерия,2024.-144с.

# Приложение 1 Листинг программы

Программа выполнена в системе MATLAB. Графики сформированы с помощью встроенных возможностей системы.

klines = 0;

**while** (klines <= 0)

klines = input("Введите кол-во килострок:\n");

**end**

% чтение данных проектов

data = read\_data('COCOMO\_data.txt');

% количество проектов, на которых проходит обучение

learning\_projects\_num = 40;

learning\_data\_indexes = randperm(length(data), learning\_projects\_num);

learning\_data = cell(1, learning\_projects\_num);

check\_data = cell(1, length(data) - learning\_projects\_num);

learn\_cnt = 1;

check\_cnt = 1;

**for** i = 1:length(data)

**if** (any(ismember(learning\_data\_indexes, i)))

learning\_data{learn\_cnt} = data{i};

learn\_cnt = learn\_cnt + 1;

**else**

check\_data{check\_cnt} = data{i};

check\_cnt = check\_cnt + 1;

**end**

**end**

% % данные для обучения - 40 шт

% learning\_data = {data{1:20}, data{41:60}};

%

% % данные для проверки - 20 шт

% check\_data = {data{21:40}};

f = @(x)fitness(x(1), x(2), learning\_data);

cross = @(a, b) flat\_crossover(a, b);

res = float\_genetic(f, 100, 2, 20, 0.5, 0.1, cross, 1, 100, 0, 1);

a = res{**end**}{1}(1);

b = res{**end**}{1}(2);

costs = efm(a, b, klines);

fprintf("Затраты на проект размером %f килострок: %f человекомесяцев\n", klines, costs);

errors = zeros(1, length(res));

**for** i = 1:length(res)

errors(i) = res{i}{2};

**end**

figure("Name", "Уменьшение ошибки на каждом этапе");

plot(errors, "-o");

legend("Ошибка ED - Евклидово расстояние");

learn\_data\_errors = zeros(length(learning\_data), 2);

nums = strings(length(learning\_data), 1);

**for** i = 1:length(learning\_data)

s = learning\_data{i};

learn\_data\_errors(i, 1) = ed(s.EF\_C, s.EF);

learn\_data\_errors(i, 2) = ed(efm(a,b,s.L), s.EF);

nums(i) = string(s.num);

**end**

figure("Name", "Ошибки для обучающих данных");

bar(nums, learn\_data\_errors);

legend("Ошибка ED COCOMO", "Ошибка ED ГА");

xlabel("Номера проектов");

ylabel("Ошибка ED");

yscale log

check\_data\_errors = zeros(length(check\_data), 2);

nums = strings(length(check\_data), 1);

**for** i = 1:length(check\_data)

s = check\_data{i};

check\_data\_errors(i, 1) = ed(s.EF\_C, s.EF);

check\_data\_errors(i, 2) = ed(efm(a,b,s.L), s.EF);

nums(i) = string(s.num);

**end**

figure("Name", "Ошибки для тестовых данных");

bar(nums, check\_data\_errors);

legend("Ошибка ED COCOMO", "Ошибка ED ГА");

xlabel("Номера проектов");

ylabel("Ошибка ED");

yscale log

check\_error = fitness(a, b, check\_data);

% for i = 1:length(check\_data)

% s = check\_data

% error = f(a, b, )

% end

% s = fitness(1, 1, learning\_data);

% float\_genetic( ...

% f, ...

% population\_size, ...

% nodes\_count, ...

% max\_steps, ...

% cross\_prob, ...

% mutation\_prob, ...

% cross, ...

% coord\_from, ...

% coord\_to)

% Вариант 4: ГА Веществ. вектор ED арифметич арифметич рулетка

% Фитнесс - функция: эвклидово расстояние

**function** **v** = **fitness**(a, b, learn\_data)

len = length(learn\_data);

sum = 0;

**for** i = 1:len

s = learn\_data{i};

ef = s.EF;

l = s.L;

efmi = efm(a, b, l);

subsum = (ef - efmi) .^ 2;

sum = sum + subsum;

**end**

v = sqrt(sum ./ len);

**end**

**function** **v** = **ed**(ef, efmi)

len = length(ef);

sum = 0;

**for** i = 1:len

subsum = (ef(i) - efmi(i)) .^ 2;

sum = sum + subsum;

**end**

v = sqrt(sum ./ len);

**end**

% Расчет сложности проекта по кол-ву строк,

% коэффициентам a и b и кол-ву килострок l

**function** **e** = **efm**(a, b, l)

e = (l .^ b) .\* a;

**end**

% Чтение файла с данными для обучения

**function** **data** = **read\_data**(fileName)

data = {};

fileID = fopen(fileName, 'r');

formatSpec = '%f';

a = fscanf(fileID, formatSpec);

row\_len = 7;

**for** i = 1:((length(a)/row\_len))

offset = row\_len \* (i - 1);

s.num = a(offset + 1);

s.L = a(offset + 2);

s.EF = a(offset + 3);

s.EF\_C = a(offset + 4);

data{length(data) + 1} = s;

**end**

**end**

**function** **res** = **float\_genetic**( ...

f, ...

population\_size, ...

nodes\_count, ...

max\_steps, ...

cross\_prob, ...

mutation\_prob, ...

cross, ...

a\_coord\_from, ...

a\_coord\_to, ...

b\_coord\_from, ...

b\_coord\_to)

population = generate\_population( ...

population\_size, ...

nodes\_count, ...

a\_coord\_from, ...

a\_coord\_to, ...

b\_coord\_from, ...

b\_coord\_to);

res = cell(1, max\_steps);

**for** i = 1:max\_steps

**if** (length(population) < population\_size)

a=1;

**end**

parents = reproduction(population, f);

% Сохраняем лучшую особь

best = best\_individ(f, population);

best\_ind = population{best};

**if** (best > 1)

start = population(1:best-1);

**else**

start = cell(0,1);

**end**

**if** (best < length(population))

ending = population(best+1:**end**);

**else**

ending = cell(0,1);

**end**

without\_best\_pop = [start;ending];

% рождение и мутация детей

children = crossover(parents, cross\_prob, cross);

children = mutation(children, mutation\_prob);

% редукция

new\_population = [without\_best\_pop; children];

population = reduction(f, new\_population, population\_size - 1);

population = [{best\_ind}; population];

best = best\_individ(f, population);

best\_ind = population{best};

fprintf("Шаг %d: лучшие значения:%s:%f\n", i, mat2str(best\_ind), f(best\_ind));

% fprintf("Длина маршрута: %d\n", f(best\_ind));

res{i} = {best\_ind, f(best\_ind)};

**end**

% best = best\_individ(f, population);

% res = {population{best}; f(population{best})};

**end**

**function** **num** = **best\_individ**(f, pop)

min\_val = intmax();

num = 0;

**for** i = 1:length(pop)

val = f(pop{i});

**if** (val < min\_val)

min\_val = val;

num = i;

**end**

**end**

**end**

% Плоский кроссовер

**function** **child** = **flat\_crossover**(p1, p2)

dim = length(p1);

child = zeros(dim);

**for** i = 1:dim

diff = abs(p1(i) - p2(i));

gap = diff .\* 0.25;

child(i) = rand\_range(min(p1(i), p2(i)) - gap, max(p2(i), p1(i)) + gap);

**end**

**end**

% Мутация

**function** **mutated\_pop** = **mutation**(pop, prob)

**for** i = 1:length(pop)

**if** (rand() < prob)

val = pop{i};

**for** j = 1:length(val)

val(j) = rand\_range(val(j) - 1, val(j) + 1);

**end**

pop{i} = val;

**end**

**end**

mutated\_pop = pop;

**end**

% function draw\_route(path, coords)

% x = zeros(1, length(path));

% y = zeros(1, length(path));

%

% for i = 1:length(path)

% node = path(i);

% x(i) = coords(node, 1);

% y(i) = coords(node, 2);

% end

%

% figure;

% plot(x, y, '-x');

%

% end

% Генерация начальной популяции

**function** **pop** = **generate\_population**(pop\_size, ind\_size, from, to, from2, to2)

pop = cell(pop\_size, 1);

% pop = zeros(pop\_size, ind\_size);

**for** i = 1:pop\_size

ind = zeros(1, ind\_size);

ind(1) = rand\_range(from, to);

ind(1) = rand\_range(from2, to2);

% for j = 1:ind\_size

% ind(j) =

% end

pop{i} = ind;

**end**

**end**

% случайное число в диапазоне

**function** **ret** = **rand\_range**(from, to)

ret = (to-from) .\* rand() + from;

**end**

% Редукция

**function** **reducted\_pop** = **reduction**(f, pop, len)

valind = configureDictionary("double", "cell");

cnt = 0;

arr = {{}};

**for** i = 1:length(pop)

val = f(pop{i});

**if** cnt > 0

**if** (valind.isKey(val))

arr = valind.lookup(val);

**else**

arr = {{}};

**end**

**else**

arr = {{}};

**end**

subarr = arr{1};

subarr{length(subarr) + 1} = pop{i};

arr{1} = subarr;

% if (cnt == 0)

% val2ind = dictionary(val, arr);

% end

valind = valind.insert(val, arr);

cnt = cnt + 1;

**end**

reducted\_pop = {};

**while** (length(reducted\_pop) < len & ~isempty(valind.keys()))

minval = min(valind.keys);

arr = valind.lookup(minval);

unique = configureDictionary("double", "double");

**for** j = 1:length(arr{1})

ind = arr{1}{j};

% is\_member = ismember([ind], cell2mat(reducted\_pop));

**if** (~unique.isKey(ind))

reducted\_pop = [reducted\_pop; ind];

unique = unique.insert(ind, 1);

**end**

**end**

valind = valind.remove(minval);

**end**

**end**

% Репродукция

**function** **intermediate\_population** = **reproduction**(population, f)

pop\_size = length(population);

% Оператор репродукции

intermediate\_population = cell(pop\_size, 1);

individ\_values = zeros(pop\_size, 1);

% сумма всех значений и значение для каждой особи

**for** i = 1:pop\_size

val = f(population{i});

individ\_values(i) = val;

**end**

values\_sum = sum(individ\_values);

% все особи одинаковы

**if** values\_sum == 0

intermediate\_population = population;

**return**

**end**

potentials = zeros(pop\_size, 1);

% подсчет потенциала для каждой особи

% потенциалы отрицательны, т.к. ищем минимум

**for** i = 1:length(individ\_values)

val = individ\_values(i);

prob = -(val / values\_sum);

potentials(i) = prob;

**end**

min\_potential = min(potentials);

% сдвигаем потенциалы в положительную часть

**for** i = 1:length(individ\_values)

potentials(i) = potentials(i) - min\_potential;

**end**

sum\_potentials = sum(potentials);

% выбор такого же кол-ва особей

**for** i = 1:length(population)

% крутите барабан

shot = rand\_range(0, sum\_potentials);

individ\_num = 0;

tmp\_sum = 0;

% определяем куда попали барабаном

**for** j = 1:length(population)

individ\_num = j;

tmp\_sum = tmp\_sum + potentials(individ\_num);

**if** (tmp\_sum >= shot)

**break**

**end**

**end**

% добавляем выбранную особь в промежуточную популяцию

individ = population{individ\_num};

intermediate\_population{i} = individ;

**end**

**end**

% Скрещивание

**function** **pop** = **crossover**(parents, prob, cros)

len = length(parents);

pop = cell(0, 1);

**for** i = 1:len

p1 = parents{round(rand\_range(1, len))};

p2 = parents{round(rand\_range(1, len))};

**if** (rand() < prob)

child = cros(p1, p2);

pop{length(pop) + 1, 1} = child;

**end**

**end**

**end**