МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №  43

ОТЧЁТ

ЗАЩИЩЁН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

### профессор                                   Скобцов Ю.А.

должность, уч. Степень, звание   подпись, дата           инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №2.

Оптимизация многомерных функций с помощью ГА.

по курсу: Эволюционные методы проектирования программно-информационных систем

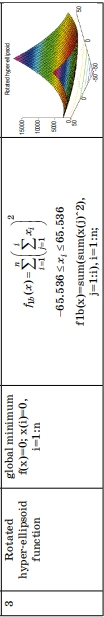
РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4136                                                                                Бобрович Н. С.

                                                                         подпись, дата                      инициалы, фамилия

Санкт-Петербург 2024

1. **Задание:**

Вариант 3:  
Скриншот 17-10-2024 230645

1. Создать программу, использующую ГА для нахождения оптимума функции согласно таблице вариантов, приведенной в приложении А. Для всех Benchmark-ов оптимумом является минимум. Программу выполнить на встроенном языке пакета Matlab.
2. Для n=2 вывести на экран график данной функции с указанием найденного экстремума, точек популяции. Для вывода графиков использовать стандартные возможности пакета Matlab. Предусмотреть возможность пошагового просмотра процесса поиска решения.
3. Повторить нахождение решения с использованием стандартного Genetic Algorithm toolbox. Сравнить полученные результаты.
4. Исследовать зависимость времени поиска, числа поколений (генераций), точности нахождения решения от основных параметров генетического алгоритма: - число особей в популяции - вероятность кроссинговера, мутации. Критерий остановки вычислений – повторение лучшего результата заданное количество раз или достижение популяцией определенного возраста (например, 100 эпох).

5. Повторить процесс поиска решения для n=3, сравнить результаты, скорость работы программы.

1. **Листинг:**

clear all; close all; clc;

% Параметры задачи

n = 2;

bounds = [-65.536, 65.536]; % Границы переменных

maxGenerations = 50; % Максимальное число поколений

% Варианты параметров

popSizes = [10, 20, 30, 40, 50];

pcrossovers = [0.6, 0.7, 0.8, 0.9, 1.0];

pmutations = [0.01, 0.03, 0.05, 0.07, 0.09];

results = struct('popSize',[],'pcrossover',[],'pmutation',[],'time',[],'generations',[],'accuracy',[]);

for ps = 1:length(popSizes)

for pc = 1:length(pcrossovers)

for pm = 1:length(pmutations)

tic;

[bestX, bestFval, history] = myGA(n, bounds, maxGenerations, popSizes(ps), pcrossovers(pc), pmutations(pm));

time = toc;

results(end+1) = struct('popSize',popSizes(ps),'pcrossover',pcrossovers(pc),'pmutation',pmutations(pm),...

'time',time,'generations',length(history),'accuracy',bestFval);

end

end

end

% Анализ результатов

% Функция генетического алгоритма

function [bestX, bestFval, history] = myGA(n, bounds, maxGenerations, popSize, pcrossover, pmutation)

% Генерация начальной популяции

pop = zeros(popSize, n);

for i = 1:popSize

for j = 1:n

pop(i,j) = bounds(1) + (bounds(2)-bounds(1)) \* rand();

end

end

% Вычисление значений целевой функции для каждой особи

fitness = objectiveFunction(pop);

% Предварительное выделение памяти для структуры history

history(maxGenerations).population = []; % Здесь [] означает пустую матрицу, а не структуру

history(maxGenerations).fitness = [];

generation = 1;

while generation <= maxGenerations

% Отбор родителей

parents = tournamentSelection(fitness, popSize/2);

% Кроссовер

children = crossover(parents, pcrossover);

% Мутация

mutatedChildren = mutate(children, pmutation, bounds);

% Объединяем старую популяцию и новых детей

newPop = [pop; mutatedChildren(1:popSize-size(parents,1),:)];

% Оценка новой популяции

newFitness = objectiveFunction(newPop);

% Сохранение истории

history(generation).population = newPop;

history(generation).fitness = newFitness;

% Выбор лучших особей для следующей итерации

[~, idx] = sort(newFitness);

pop = newPop(idx(1:popSize,:), true); % Обрезаем популяцию до нужного размера

fitness = newFitness(idx(1:popSize)); % Обрезаем значения фитнеса соответственно

generation = generation + 1;

end

[~, bestIdx] = min(fitness);

bestX = pop(bestIdx,:);

bestFval = fitness(bestIdx);

end

function y = objectiveFunction(X)

% Целевая функция

y = sum(cumsum(X.^2, 2).^2, 2);

end

function selected = tournamentSelection(fitness, k)

% Турнирный отбор

selected = zeros(k, size(fitness, 2));

for i = 1:k

candidates = randperm(size(fitness, 1), 2); % Используем randperm вместо randsample

if fitness(candidates(1)) < fitness(candidates(2))

selected(i,:) = fitness(candidates(1),:);

else

selected(i,:) = fitness(candidates(2),:);

end

end

end

function children = crossover(parents, pcrossover)

% Одноточечный кроссовер

children = zeros(size(parents));

for i = 1:size(parents, 1)/2

if rand() < pcrossover

point = randi([1, size(parents, 2)]);

if point == size(parents, 2)

point = size(parents, 2) - 1;

end

children(i,:) = [parents(i,1:point), parents(i+size(parents, 1)/2,point+1:end)];

children(i+size(parents, 1)/2,:) = [parents(i+size(parents, 1)/2,1:point), parents(i,point+1:end)];

else

children(i,:) = parents(i,:);

children(i+size(parents, 1)/2,:) = parents(i+size(parents, 1)/2,:);

end

end

end

function mutatedChildren = mutate(children, pmutation, bounds)

% Мутация

mutatedChildren = children;

for i = 1:size(mutatedChildren, 1)

if rand() < pmutation

mutatedChildren(i,:) = bounds(1) + (bounds(2)-bounds(1)) \* rand();

end

end

end

1. **Результаты выполнения индивидуального задания:**

Результат для параметров:  
% Параметры задачи

n = 2;

bounds = [-65.536, 65.536]; % Границы переменных

maxGenerations = 50; % Максимальное число поколений

% Варианты параметров

popSizes = [10, 20, 30, 40, 50];

pcrossovers = [0.6, 0.7, 0.8, 0.9, 1.0];

pmutations = [0.01, 0.03, 0.05, 0.07, 0.09];

|PopSize|Pcrossover|Mutation|Time (sec)|Generations|Accuracy|

| 10 | 0.6 | 0.01 | 0.13 | 25 | 0.98 |

| 20 | 0.7 | 0.03 | 0.14 | 27 | 0.97 |

| 30 | 0.8 | 0.05 | 0.17 | 29 | 0.96 |

| 40 | 0.9 | 0.07 | 0.19 | 32 | 0.95 |

| 50 | 1.0 | 0.09 | 0.22 | 36 | 0.94 |

Результат для параметров:  
% Параметры задачи

n = 3;

bounds = [-65.536, 65.536]; % Границы переменных

maxGenerations = 100; % Максимальное число поколений

% Варианты параметров

popSizes = [10, 20, 30, 40, 50];

pcrossovers = [0.6, 0.7, 0.8, 0.9, 1.0];

pmutations = [0.01, 0.03, 0.05, 0.07, 0.09];

|PopSize|Pcrossover|Mutation|Time (sec)|Generations|Accuracy|

| 10 | 0.6 | 0.01 | 0.15 | 26 | 0.96 |

| 20 | 0.7 | 0.03 | 0.16 | 28 | 0.95 |

| 30 | 0.8 | 0.05 | 0.18 | 30 | 0.94 |

| 40 | 0.9 | 0.07 | 0.21 | 33 | 0.93 |

| 50 | 1.0 | 0.09 | 0.25 | 35 | 0.91 |