**Федеральное государственное бюджетное образовательное учреждение высшего образования "Белгородский государственный технологический университет им. В.Г. Шухова"**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем.

**Лабораторная работа №5**

Решение оптимизационных задач эволюционно-генетическими алгоритмами.

Выполнил:

Студент группы КБ-211

Коренев Д.Н.

Принял:

Твердохлеб В.В

Оглавление

[Задание 3](#_Toc184274101)

[Выполнение 4](#_Toc184274102)

[Блок-схема алгоритма 4](#_Toc184274103)

[Листинг программы 4](#_Toc184274104)

[Результаты 8](#_Toc184274105)

*Цель работы:* Изучение особенностей генетических алгоритмов необходимых при решении оптимизационных задач.

Задание

1. Изучить теоретический материал по эволюционно-генетическим алгоритмам и их использованию при решении оптимизационных задач
2. Разработать программное приложение под управлением ОС Windows, позволяющее решить задачу комивояжера для количества вершин графа не менее 20. Интерфейс приложения должен обеспечивать визуализацию процесса поиска кратчайшего пути, а также ввод параметров работы алгоритма.
3. Ответить на контрольные вопросы.

Выполнение

1.Разработать программное приложение под управлением ОС Windows, позволяющее решить задачу комивояжера для количества вершин графа не менее 20. Интерфейс приложения должен обеспечивать визуализацию процесса поиска кратчайшего пути, а также ввод параметров работы алгоритма.

Блок-схема алгоритма

Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Автоматически созданное описание

Листинг программы

Python 3.12

import tkinter as tk

from tkinter import messagebox

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import numpy as np

import random

class TravelingSalesmanApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Задача коммивояжера")

        # Элементы интерфейса

        tk.Label(

            root, text="Размер популяции:"

        ).grid(row=0, column=0, sticky="w")

        self.population\_size\_entry = tk.Entry(root)

        self.population\_size\_entry.grid(row=0, column=1)

        self.population\_size\_entry.insert(0, "100")

        tk.Label(

            root, text="Число поколений:"

        ).grid(row=1, column=0, sticky="w")

        self.generations\_entry = tk.Entry(root)

        self.generations\_entry.grid(row=1, column=1)

        self.generations\_entry.insert(0, "200")

        tk.Label(

            root, text="Вероятность мутации (0-1):"

        ).grid(row=2, column=0, sticky="w")

        self.mutation\_rate\_entry = tk.Entry(root)

        self.mutation\_rate\_entry.grid(row=2, column=1)

        self.mutation\_rate\_entry.insert(0, "0.1")

        self.start\_button = tk.Button(

            root, text="Запустить", command=self.run\_algorithm

        )

        self.start\_button.grid(row=3, column=0, columnspan=2, pady=10)

        # Поле для визуализации

        self.figure, self.ax = plt.subplots(figsize=(5, 5))

        self.canvas = FigureCanvasTkAgg(self.figure, root)

        self.canvas.get\_tk\_widget().grid(row=4, column=0, columnspan=2)

        # Генерация графа

        self.num\_nodes = 20

        self.coords = self.generate\_graph()

    def generate\_graph(self):

        """Генерация случайных координат вершин графа."""

        return np.random.rand(self.num\_nodes, 2) \* 100

    def distance(self, a, b):

        """Расчет евклидового расстояния между двумя точками."""

        return np.linalg.norm(a - b)

    def fitness(self, route):

        """Функция приспособленности: длина маршрута."""

        return sum(

            self.distance(self.coords[route[i]], self.coords[route[i + 1]])

            for i in range(len(route) - 1)

        ) + self.distance(self.coords[route[-1]], self.coords[route[0]])

    def initialize\_population(self, population\_size):

        """Инициализация популяции маршрутов."""

        return [

            random.sample(range(self.num\_nodes), self.num\_nodes)

            for \_ in range(population\_size)

        ]

    def select\_parents(self, population, fitness\_values):

        """Рулеточный отбор родителей."""

        total\_fitness = sum(1 / fv for fv in fitness\_values)

        probabilities = [(1 / fv) / total\_fitness for fv in fitness\_values]

        return random.choices(population, weights=probabilities, k=2)

    def crossover(self, parent1, parent2):

        """Одноточечное скрещивание."""

        cut = random.randint(1, self.num\_nodes - 2)

        child = parent1[:cut] + \

            [gene for gene in parent2 if gene not in parent1[:cut]]

        return child

    def mutate(self, route, mutation\_rate):

        """Мутация путем обмена двух случайных генов."""

        if random.random() < mutation\_rate:

            i, j = random.sample(range(self.num\_nodes), 2)

            route[i], route[j] = route[j], route[i]

    def run\_algorithm(self):

        """Запуск генетического алгоритма."""

        try:

            population\_size = int(self.population\_size\_entry.get())

            generations = int(self.generations\_entry.get())

            mutation\_rate = float(self.mutation\_rate\_entry.get())

        except ValueError:

            messagebox.showerror("Ошибка", "Введите корректные параметры.")

            return

        # Инициализация

        population = self.initialize\_population(population\_size)

        best\_route = None

        best\_distance = float("inf")

        progress = []

        for generation in range(generations):

            fitness\_values = [self.fitness(route) for route in population]

            best\_gen\_route = population[np.argmin(fitness\_values)]

            best\_gen\_distance = min(fitness\_values) # type: ignore

            # Обновление глобального лучшего решения

            if best\_gen\_distance < best\_distance:

                best\_distance = best\_gen\_distance

                best\_route = best\_gen\_route

            # Обновление графика

            self.update\_plot(best\_route, best\_distance, generation)

            progress.append(best\_distance)

            # Новое поколение

            new\_population = []

            for \_ in range(population\_size):

                parent1, parent2 = self.select\_parents(

                    population, fitness\_values)

                child = self.crossover(parent1, parent2)

                self.mutate(child, mutation\_rate)

                new\_population.append(child)

            population = new\_population

        # Финальный результат

        self.update\_plot(best\_route, best\_distance, "Итог")

        messagebox.showinfo("Готово", f"Лучший маршрут: {best\_distance:.2f}")

    def update\_plot(self, route, distance, generation):

        """Обновление графика."""

        self.ax.clear()

        self.ax.scatter(self.coords[:, 0], self.coords[:, 1], color="red")

        path = [self.coords[city] for city in route] + [self.coords[route[0]]]

        self.ax.plot([p[0] for p in path], [p[1] for p in path], color="blue")

        self.ax.set\_title(f"Поколение: {generation}, Длина: {distance:.2f}")

        self.canvas.draw()

if \_\_name\_\_ == "\_\_main\_\_":

    root = tk.Tk()

    app = TravelingSalesmanApp(root)

    root.mainloop()

Результаты

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

2.Ответить на контрольные вопросы.

1. Как можно назвать на языке генетики значение свойства или вариант свойства?  
На языке генетики значение свойства называется аллель, а само свойство – ген.

2. В чем заключается важное значение функции приспособленности или функции оценки?  
Функция приспособленности определяет, насколько "хорошо" данное решение соответствует условиям задачи. Она является ключевым элементом, определяющим выживаемость индивидов.

3. Из каких основных шагов состоит классический генетический алгоритм?

* Инициализация популяции.
* Оценка функции приспособленности.
* Селекция лучших особей.
* Генерация нового поколения через скрещивание и мутацию.
* Проверка условий завершения.

4. Назовите наиболее популярный метод селекции. Опишите его особенности.  
Самый популярный метод – рулеточный отбор. Особенность в том, что вероятность выбора особи пропорциональна её значению функции приспособленности.

5. Опишите основные этапы процесса скрещивания.

* Выбор двух родительских особей.
* Определение точки разрыва хромосом.
* Обмен частями генетического материала между родителями.
* Формирование двух новых потомков.

**Вывод:** в результате выполнения лабораторной работы были изучены особенности генетических алгоритмов необходимых при решении оптимизационных задач. Генетический алгоритм показал хорошую эффективность в решении задачи коммивояжера. Оптимальные параметры алгоритма зависят от задачи: увеличенный размер популяции и число поколений улучшают результаты, но увеличивают время выполнения.