

**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 |
| Выполнение | 4 |
| Блок-схема алгоритма..... | 4 |
| Листинг программы | 4 |
| Результаты..... | 8 |

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

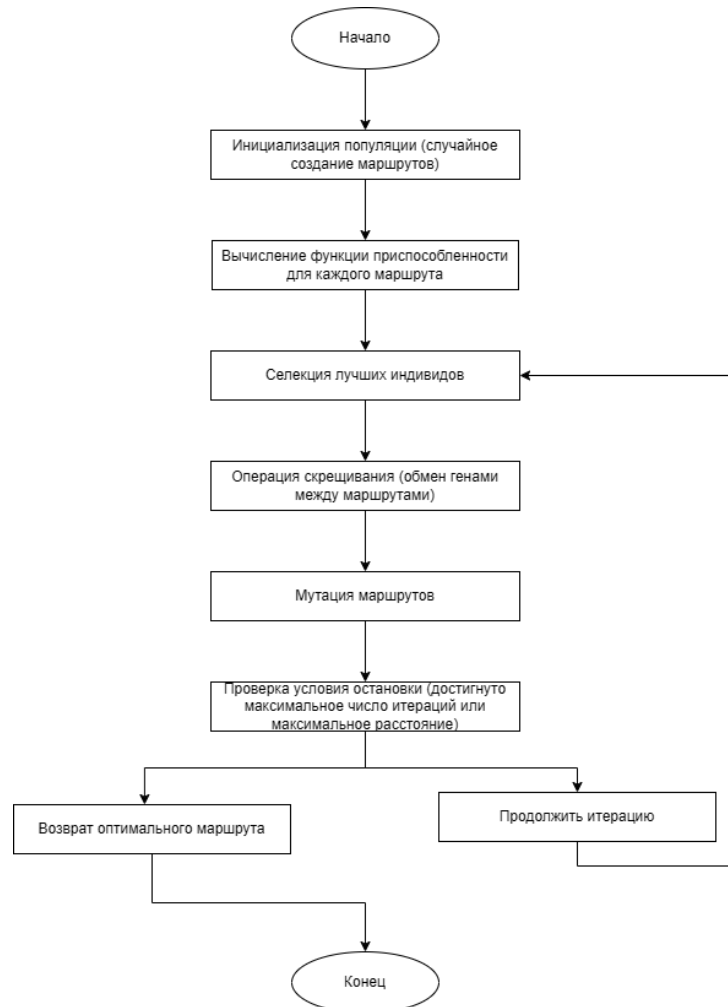
Задание

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 коммивояжера 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 коммивояжера 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



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 евклидова 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):
    """Пулеточный 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

```

```

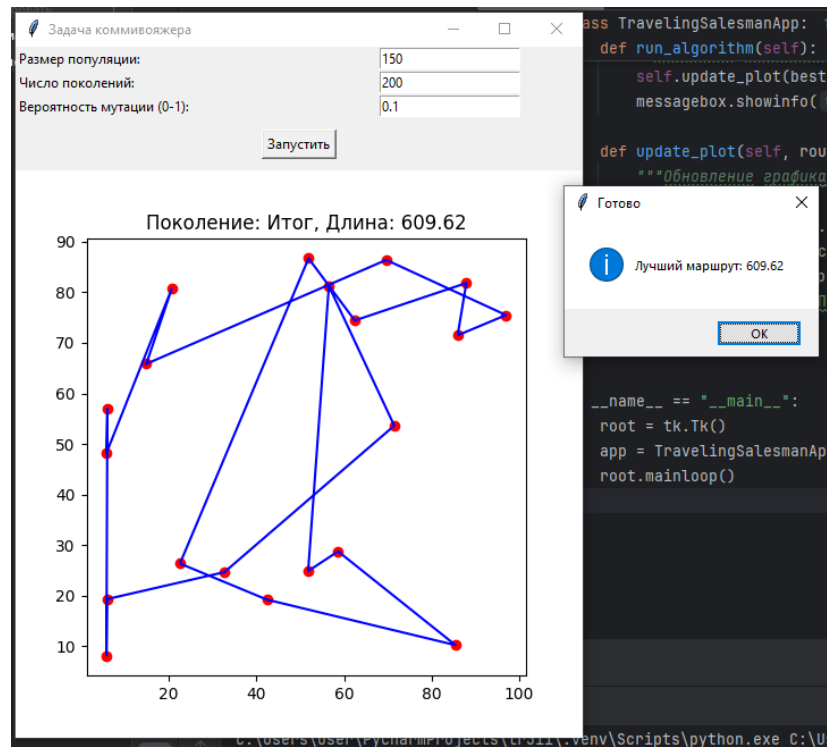
# Финальный результат
self.update_plot(best_route, best_distance, "Итог")
messagebox.showinfo("Готово", f"Лучший маршрут: {best_distance:.2f}")

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



| Размер популяции | Вероятность мутации | Число поколений | Лучший результат | Время выполнения |
|------------------|---------------------|-----------------|------------------|------------------|
| 50 | 0.1 | 200 | 494.58 | 1.2 |
| 73 | 0.1 | 200 | 535.96 | 1.93 |
| 97 | 0.1 | 200 | 560.62 | 2.71 |
| 121 | 0.1 | 200 | 598.75 | 3.73 |
| 144 | 0.1 | 200 | 590.08 | 4.35 |
| 168 | 0.1 | 200 | 532.56 | 5.4 |
| 192 | 0.1 | 200 | 515.42 | 6.54 |
| 215 | 0.1 | 200 | 534.36 | 7.58 |
| 239 | 0.1 | 200 | 525.65 | 8.71 |
| 263 | 0.1 | 200 | 522.99 | 10.07 |
| 100 | 0.1 | 50 | 650.98 | 0.74 |
| 100 | 0.1 | 73 | 641.84 | 1.07 |
| 100 | 0.1 | 97 | 628.5 | 1.36 |
| 100 | 0.1 | 121 | 565.8 | 1.74 |
| 100 | 0.1 | 144 | 657.79 | 2.18 |
| 100 | 0.1 | 168 | 632.52 | 2.47 |
| 100 | 0.1 | 192 | 652.24 | 2.89 |
| 100 | 0.1 | 215 | 561.61 | 3.2 |
| 100 | 0.1 | 239 | 615.84 | 3.52 |
| 100 | 0.1 | 263 | 598.57 | 3.56 |

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.