**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №5**

**по дисциплине «Генетические алгоритмы»**

**Тема: Классификация**

**Вариант 4**

| Студент гр. 1303 |  | Чубан Д.В. |
| --- | --- | --- |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2024

**Цель работы.**

Цель данной лабораторной работы состоит в изучении и освоении

генетических алгоритмов.

**Задание.**

Задача о воздушной почте. Почтальону необходимо доставить почту в N (задается пользователем) городов с заданными координатами (x,y) (вводится пользователем). Города находятся на местности 30 км. на 30 км. Известно, что каждый город нельзя посещать больше одного раза, и нужно вернуться в город вылета. Но у почтальона есть вертолет, и он может летать из любого города в любой другой по прямой. У городов есть приоритет доставки, города с большим приоритетом должны обязательно получать почту раньше городов с меньшим приоритетом (приоритеты задаются пользователем, а также у нескольких городов может быть одинаковый приоритет). А так как вертолет старый, то нежелательно, чтобы один перелет между городами был больше 5 километров. Необходимо составить маршрут для почтальона, чтобы он пролетел минимальное расстояние. Почтальон может начинать полет из любого города.

1. **Постановка задачи**
   1. Формализовать описание задачи в математическом виде.
   2. Описать представление решения в виде хромосомы. Пояснить выбранное представление и какие ограничения есть на хромосому.
   3. Разработать метрику для оценки качества решения. Пояснить разработанную метрику, и как она позволяет соблюдать мягкие и жесткие ограничения.
2. **Выполнение ГА**
   1. Выбрать операции для ГА: метод отбора, метод скрещивания, метод мутации. Обосновать выбор.
   2. Подобрать параметры ГА: размер популяции, вероятность скрещивания и мутации, количество поколений (если есть другие параметры, то и их тоже). Описать процесс подбора параметров.
   3. Провести оптимизацию и визуализировать полученное решение. *Графически отобразить решение. Например, траектория броска мяча или в виде таблицы показать расписание работы.*
   4. Построить график зависимости лучше и средней приспособленности в зависимости от поколения.
3. **Анализ ГА**
   1. Продемонстрировать случаи, когда находится точное решение без нарушения ограничений. Визуализируя как в п. 2.3.
   2. Продемонстрировать случаи, когда находится точное решение с нарушением мягких ограничений. Визуализируя как в п. 2.3.
   3. Продемонстрировать случаи, когда решение невозможно найти. Визуализируя как в п. 2.3.

**Выполнение работы.**

1. Постановка задачи
   1. Формализуем задачу в математическом виде.

Цель задачи – нахождение кратчайшего маршрута, проходящего через все города по 1 разу, заканчивающегося в начальном городе и удовлетворяющий условиям приоритетов и максимального расстояния для одного перелета.

*N* – количество городов

*xi, yi* – координаты i города

*сij* – расстояние между i и j городами

*pi* – приоритет доставки в i город

*fij* – 1, если ребро использовано в итоговом пути, 0, если не использовано.

Решение сводится к нахождению минимума:

(1)

С ограничениями:

Города находятся на квадрате 30x30, (0,0) – левый нижний угол, (30, 30) – правый верхний.

У каждого города есть ровно одна входящая и исходящая дуги:

Исключим недопустимые решения, создающие изолированные маршруты:

(5)

Если *pi < pj*, то *fij = 0*

Максимальное расстояние одного перелета – *cijfij ≤ 5*

* 1. Опишем представление решения в виде хромосомы.

Хромосома представляет собой последовательность городов, которые нужно посетить почтальону. При этом в в хромосоме не будет повторяющихся городов, а каждый город должен удовлетворять условиям, что город с большим приоритетом появляется в последовательности раньше города с меньшим приоритетом, а расстояние между любыми соседними городами не превышает 5 км.

* 1. Разработаем метрику для оценки качества решения.

В метрике учтем общую длину пути и ограничения.

Жесткие ограничения – города с большим приоритетом должны посещаться раньше остальных.

Мягкие ограничения – длина пути между городами не более 5 км.

Функция приспособленности, учитывающая метрику указана в листинге 1.3.1, расчет расстояния между городами в листинге 1.3.2. Штрафы подобраны для недопущения нарушения жесткого ограничения и учитывания мягкого ограничения.

Листинг 1.3.1 – функция приспособленности.

| HARD\_PENALTY = 10000  LIGHT\_PENALTY = 10  def pathFit(towns, individual):  result\_path = 0  for i in range(1, len(individual)):  if towns[individual[i]].p > towns[individual[i - 1]].p:  result\_path += HARD\_PENALTY  distance = count\_distance(towns[individual[i]], towns[individual[i - 1]])  if distance > 5:  result\_path += LIGHT\_PENALTY  result\_path += distance  distance = count\_distance(towns[individual[-1]], towns[individual[0]])  if distance > 5:  result\_path += LIGHT\_PENALTY  result\_path += distance  return result\_path, |
| --- |

Листинг 1.3.2 – функция расчета расстояния между городами.

| def count\_distance(town1, town2):  return math.sqrt((town1.x - town2.x) \*\* 2 + (town1.y - town2.y) \*\* 2) |
| --- |

1. Выполнение генетического алгоритма
   1. Выберем операции для генетического алгоритма

Метод отбора – турнирный отбор, позволит решить проблему отбора наименее приспособленных особей с сохранением разнообразия популяции и медленнее уходить в локальный минимум.

Метод скрещивания – упорядоченное скрещивание, т.к. города в хромосоме не должны повторяться.

Метод мутации – мутация перетасовкой, позволит разнообразить получаемые мутации.

* 1. Подберем параметры генетического алгоритма

Вер-ть скрещивания – 0.9, стандартно

Вер-ть мутации – 0.1, стандартно

Зал славы – 3, чтобы получить 3 лучших решения

Для остальных параметров проведем исследование на 10 городах со случайными параметрами (листинг 2.2.1)

Листинг 2.2.1 – создание 10 городов со случайными параметрами.

| towns = [Town(random.uniform(0, 30), random.uniform(0, 30), random.randint(0, 30), i) for i in range(MAX\_TOWNS)]  toolbox = base.Toolbox()  creator.create("FitnessMin", base.Fitness, weights=(-1.0,))  creator.create("Individual", list, fitness=creator.FitnessMin)  toolbox.register("createTowns", random.sample,  range(MAX\_TOWNS), MAX\_TOWNS)  toolbox.register("individualCreator", tools.initIterate,  creator.Individual, toolbox.createTowns)  toolbox.register("populationCreator", tools.initRepeat, list,  toolbox.individualCreator)  toolbox.register("evaluate", pathFit, towns) |
| --- |

Создадим список возможных значений для исследуемых параметров и выберем дающие лучший результат (листинг 2.2.2)

Листинг 2.2.2 – нахождение лучших параметров.

| population\_sizes = [50, 100, 200]  max\_generation\_values = [50, 100, 200]  indpbs = [0.01, 0.05, 0.1]  tournsizes = [2, 3, 5]  best\_result = float('inf')  best\_individual = None  best\_params = None  for population\_size, max\_generations, indpb, tournsize in itertools.product(population\_sizes, max\_generation\_values, indpbs, tournsizes):  toolbox.register("select", tools.selTournament, tournsize=tournsize)  toolbox.register("mate", tools.cxOrdered)  toolbox.register("mutate", tools.mutShuffleIndexes, indpb=indpb)  stats = tools.Statistics(lambda ind: ind.fitness.values)  stats.register("min", np.min)  stats.register("avg", np.mean)  population = toolbox.populationCreator(n=population\_size)  hof = tools.HallOfFame(maxsize=HALL\_OF\_FAME\_SIZE)  population, logbook = algorithms.eaSimple(population=population, toolbox=toolbox, cxpb=CROSSOVER\_PROB, mutpb=MUTATION\_PROB, ngen=max\_generations, halloffame=hof, verbose=False, stats=stats)  current\_best = hof.items[0].fitness.values[0]  if current\_best < best\_result:  best\_result = current\_best  best\_individual = hof.items[0]  best\_logbook = logbook  best\_params = (population\_size, max\_generations, indpb, tournsize)  print("Best Params = ", best\_params) |
| --- |

Получены следующие параметры:

Размер популяции – 50

Кол-во поколений – 50

Вероятность каждого гена поменять позицию – 0.01

Размер турнира – 5

Полученное лучшее решение – [9, 7, 5, 6, 8, 0, 1, 3, 2, 4]

* 1. Оптимизируем и визуализируем решение (листинг 2.3, рис. 2.3)

Листинг 2.3 – оптимизация и визуализация решения.

| x\_coords = [towns[i].x for i in best\_individual] +[towns[best\_individual[0]].x]  y\_coords = [towns[i].y for i in best\_individual] +[towns[best\_individual[0]].y]  plt.figure(figsize=(10, 10))  plt.plot(x\_coords, y\_coords, 'o-', markersize=8, label='Route')  for i, town in enumerate(best\_individual):  plt.annotate(f"Town{str(town)}, p={towns[town].p:.2f}", (towns[town].x, towns[town].y), textcoords="offset points", xytext=(0,10), ha='center', color="black")  for i in range(len(best\_individual)):  town1 = towns[best\_individual[i]]  town2 = towns[best\_individual[(i + 1) % len(best\_individual)]]  mid\_x = (town1.x + town2.x) / 2  mid\_y = (town1.y + town2.y) / 2  plt.text(mid\_x, mid\_y, f"dist:{count\_distance(town1, town2):.2f}", color="black", fontsize=8)  plt.title("Best Route")  plt.xlabel("X")  plt.ylabel("Y")  plt.show() |
| --- |

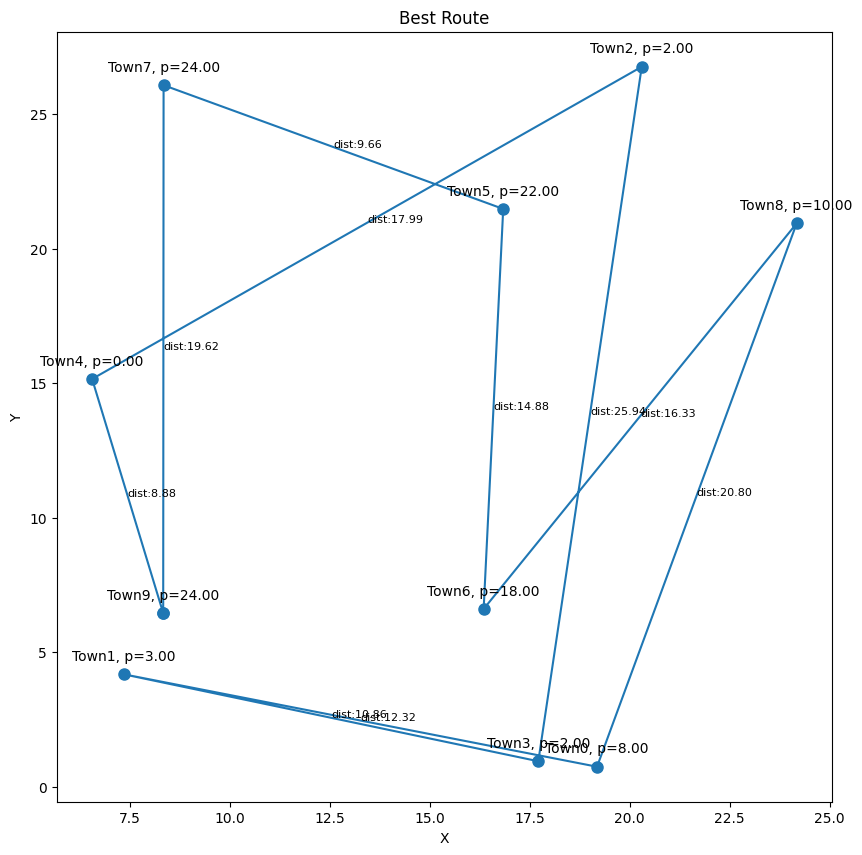


Рисунок 2.3 – визуализация лучшего решения.

Из рисунка 2.3 видно, что жесткое ограничение соблюдается, а мягкое – нет. Скорее всего, все города удалены друг от друга более, чем на 5 км.

* 1. Построим график зависимости лучшей и средней приспособленности в зависимости от поколения (листинг 2.4, рис. 2.4)

Листинг 2.4 – график зависимости лучшей и средней приспособленности в зависимости от поколения.

| minFitnessValues, meanFitnessValues = best\_logbook.select("min", "avg")  plt.plot(minFitnessValues, color='blue', label="min")  plt.plot(meanFitnessValues, color='orange', label="avg")  plt.xlabel('Generation')  plt.ylabel('Fitness')  plt.title('Min and Average Fitness over Generations')  plt.legend()  plt.show() |
| --- |

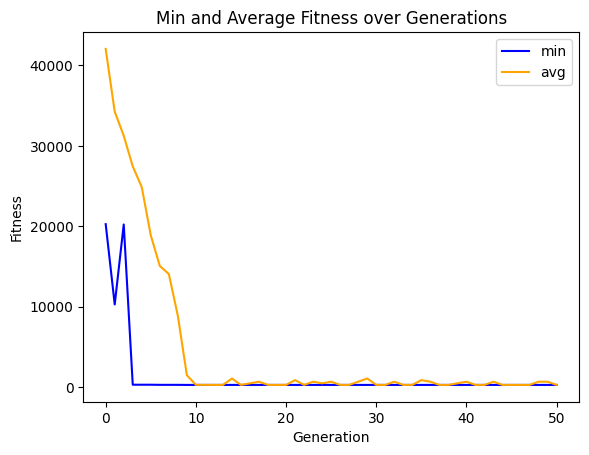


Рисунок 2.4 – график зависимости лучшей и средней приспособленности в зависимости от поколения

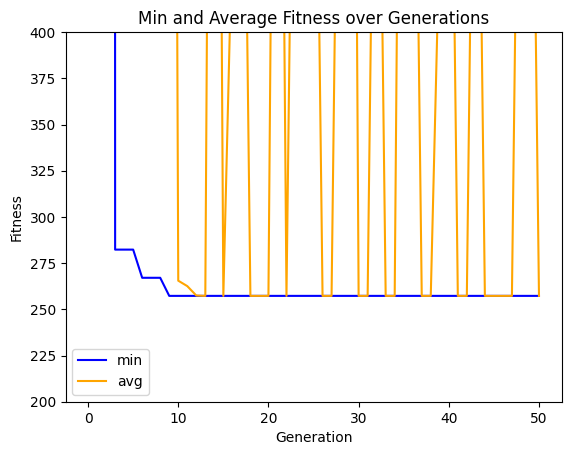


Рисунок 2.4.1 – график зависимости лучшей и средней приспособленности в зависимости от поколения в приближении.

Из рисунков 2.4 и 2.4.1 можно увидеть, что минимальное значение получается примерно на 9 поколении. Полученный минимум является суммой штрафов и длины пути. Колебания значений приспособленности скорее всего показывают попадания алгоритма в локальные оптимумы, из которых он затем выходит, находя далее лучшее решение.

**Выводы.**

В лабораторной работе была решена задача доставки почты во множество городов с использованием генетического алгоритма. Были наложены ограничения в виде приоритета доставки почты (жесткое ограничение) и ограниченности пути из одного города в другой (мягкое ограничение).

Хромосома представляла из себя порядковый список неповторяющихся городов, т.е. путь почтальона.

Операции генетического алгоритма включали в себя турнирный отбор, упорядоченное скрещивание и перестановочную мутацию.

Метрика качества оценивала длину пройденного пути с учетом мягких и жестких условий.

Параметры алгоритма подбирались с целью уменьшения итогового пути и повышения общего качества решения.

Полученный оптимальный маршрут был визуализирован.