МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Институт Информационных Технологий
Кафедра Вычислительной Техники

Практическая работа №4
«Муравьиный алгоритм»

по дисциплине
«Системный анализ данных СППР»

Студент группы: ИКБО-04-22 — Егоров Л.А.
(Ф.И.О. студента)

Принял — Железняк Л.М.
(Ф.И.О. преподавателя)

Москва 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Муравьиный алгоритм — это эвристический метод оптимизации, разработанный итальянским ученым Марко Дориго в 1992 году. Работа алгоритма вдохновлена тем, как колония муравьев отправляется на поиски пищи. Каждый муравей оставляет на своем пути феромоны — химические вещества, привлекающие других муравьев. Чем больше муравьев проходит по определенному пути, тем сильнее концентрация феромонов на нем. В результате, большинство муравьев выбирает путь с наибольшей концентрацией феромонов, который, как правило, является кратчайшим.

Муравьиный алгоритм моделирует это поведение. Вместо реальных муравьев и феромонов, алгоритм использует «искусственных муравьев» и «искусственные феромоны».

Искусственные муравьи перемещаются по графу, представляющему пространство поиска, и оставляют феромоны на ребрах, которые они посещают. Вероятность выбора муравьем определенного ребра зависит от концентрации феромонов на нем и других факторов, таких как расстояние. Со временем, феромоны испаряются, что позволяет алгоритму «забывать» неудачные пути и сосредотачиваться на наиболее перспективных.

Муравьиный алгоритм широко используется для решения различных задач оптимизации, таких как:

- задача коммивояжера: поиск кратчайшего маршрута, проходящего через все заданные города;
- распределение ресурсов: оптимальное распределение ресурсов (например, рабочей силы, оборудования) для выполнения задач;
- планирование: составление расписаний, оптимизация маршрутов транспорта;
- сетевое проектирование: поиск оптимальных путей в сетях (например, компьютерных, транспортных).

1 АЛГОРИТМ РОЯ ЧАСТИЦ

1.1 Описание алгоритма

Сначала происходит инициализация начальных параметров и самой муравьиной колонии — для алгоритма муравьи создаются в количестве, равном количеству вершин в графе, и каждый из них начинает свой путь со своей вершины.

Важно отметить, что на каждую дугу графа «наносят» феромон — число, сгенерированное псевдослучайным образом в интервале от 0 до 1. Оно является одинаковым для всех дуг перед начальной итерацией.

Далее происходит построение пути для каждого муравья в колонии. Выбор муравьём новой вершины определяется с помощью вероятности, определяемой по Формуле 1.1.1.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{u \in N_i^k} t_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)}, \text{если } j \in N_i^k \\ 0, \text{если } j \notin N_i^k \end{cases}, \quad (1.1.1)$$

где

```
(
  equation(block: false, body: [i]),
  [номер текущей вершины муравья],
)

;

(
  equation(block: false, body: [j]),
  [номер вершины, куда муравей может перейти],
)

;

(
  equation(
    block: false,
    body: attach(base: [τ], b: sequence([i], [ ], [j])),
  ),
  [количество феромона на дуге],
)

;
```

```
(
  equation(
    block: false,
    body: attach(base: [η], b: sequence([i], [ ], [j])),
  ),
  [априорная эффективность перехода по дуге из i в j],
)

;

(
  equation(
    block: false,
    body: attach(base: [N], t: [k], b: [i]),
  ),
  [множество доступных вершин для перемещения],
)

;

(
  equation(block: false, body: sequence([α], [,], [ ], [β])),
  sequence(
    [свободные параметры алгоритма],
    [ ],
    [(вес фермента и коэффициент эвристики)],
  ),
)

.
```

После построения пути для каждого муравья высчитывается длина его пути. Далее на каждой дуге происходит испарение феромона (1.1.2).

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) \ , \qquad (1.1.2)$$

где

```
(
  equation(block: false, body: [ρ]),
  sequence(
    [коэффициент испарения,],
    [ ],
    equation(
      block: false,
      body: sequence(
        [ρ],
        [ ],
        [∈],
        [ ],
        lr(body: sequence([[], [0], [,], [1], []])),
      ),
    ),
  ),
)

.
```

Для каждой дуги происходит изменение феромона в зависимости от того, насколько оптимальный путь получился у муравьёв (1.1.3).

$$\tau_{ij}(t+1) = t_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}(t) \qquad (1.1.3)$$

где

```
(
  sequence(
    v(amount: 12pt),
    styled(child: equation(
      block: true,
      numbering: none,
      body: equation(
        block: false,
        body: sequence(
          [Δ],
          [ ],
          attach(base: [τ], t: [k], b: sequence([i], [ ], [j])),
          [ ],
          lr(body: sequence([(], [t], [)])),
          [ ],
          [=],
          [ ],
          cases(
            children: (
              sequence(
                frac(num: [Q], denom: attach(base: [L], t: [k])),
                [ ],
                lr(body: sequence([(], [t], [)])),
                [ ],
                align-point(),
                [, если дуга ],
                [ ],
                lr(body: sequence([(], [i], [,], [ ], [j], [)])),
                [ ],
                [ есть в пути ],
                [ ],
                attach(base: [x], t: [k]),
                [ ],
                lr(body: sequence([(], [t], [)])),
              ),
              sequence([0], [ ], align-point(), [, иначе], [ ]),
            ),
          ),
          [ ],
          [  ],
          [ ],
          [−],
        ),
      ),
    ), ..),
  ),
  [изменение количества феромона в зависимости от длины пройденного пути],
)
```

;

```
(
  equation(block: false, body: [Q]),
  [положительная константа],
)
```

.

Точкой останова алгоритма является выполнение заданного числа итераций.

1.2 Постановка задачи

Цель работы: реализовать задачу коммивояжера муравьиным алгоритмом для нахождения приближённого оптимального маршрута.

Поставлены следующие задачи:

- изучить муравьиный алгоритм;
- выбрать предметную область для задачи коммивояжера;
- произвести ручной расчёт одной итерации алгоритма для двух муравьёв;
- разработать программную реализацию муравьиного алгоритма для задачи коммивояжера.

Условие задачи коммивояжёра: дан полный граф, т.е. из каждой вершины можно пройти в любую другую вершину. В этом графе нужно найти полный путь минимальной длины, т.е. обойти каждую вершину в графе по одному разу.

Выбранная предметная область для задачи коммивояжёра: в торговом центре расположено n магазинов. Человеку нужно пройтись по всем этим магазинам, при этом ему нужно затратить как можно меньше усилий на это, т.е. общий пройденный путь должен быть минимально возможным. Поэтому нужно определить минимальный путь, позволяющий обойти все магазины.

1.3 Ручной расчёт алгоритма

Для ручного расчёта число магазинов в задаче взято равным 6, число муравьёв — 2. Значения свободных параметров: $\alpha = 1, \beta = 5, \rho = 0.5$.

Для каждого магазина случайным образом сгенерированы координаты в двумерном пространстве (диапазон координат от -10 до 10). Эти данные представлены заданы в Таблице 1.3.1.

Таблица 1.3.1 — Характеристики магазинов

Номер магазина	**Координата по x**	**Координата по y**
1	92	-97
2	-67	28

Продолжение Таблицы 1.3.1

3	-14	-90
4	-67	72
5	19	75
6	77	-97

Для расчёта расстояний между вершинами в графе использовалось Евклидово расстояние для точек в двумерном пространстве (1.3.1).

$$d = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} \tag{1.3.1}$$

Ниже приведены расчёты длины каждого ребра в графе, т.е. рассчитаны длины путей между каждой парой вершин.

$$\sqrt{(92 + 67)^2 + (-97 - 28)^2} = 202.25$$
$$\sqrt{(92 + 14)^2 + (-97 + 90)^2} = 106.23$$
$$\sqrt{(92 + 67)^2 + (-97 - 72)^2} = 232.04$$
$$\sqrt{(92 - 19)^2 + (-97 - 75)^2} = 186.85$$
$$\sqrt{(92 - 77)^2 + (-97 + 97)^2} = 15.0$$
$$\sqrt{(-67 + 14)^2 + (28 + 90)^2} = 129.36$$
$$\sqrt{(-67 + 67)^2 + (28 - 72)^2} = 44.0$$
$$\sqrt{(-67 - 19)^2 + (28 - 75)^2} = 98.01$$
$$\sqrt{(-67 - 77)^2 + (28 + 97)^2} = 190.69$$
$$\sqrt{(-14 + 67)^2 + (-90 - 72)^2} = 170.45$$
$$\sqrt{(-14 - 19)^2 + (-90 - 75)^2} = 168.27$$
$$\sqrt{(-14 - 77)^2 + (-90 + 97)^2} = 91.27$$
$$\sqrt{(-67 - 19)^2 + (72 - 75)^2} = 86.05$$
$$\sqrt{(-67 - 77)^2 + (72 + 97)^2} = 222.03$$
$$\sqrt{(19 - 77)^2 + (75 + 97)^2} = 181.52$$

Рассчитанные длины рёбер сведены в Таблицу 1.3.2 с указанием вершин, составляющих ребро.

Таблица 1.3.2 — Длины рёбер в графе

Ребро	**Длина ребра**
0 → 1	202.25
0 → 2	106.23
0 → 3	232.04
0 → 4	186.85
0 → 5	15.00
1 → 2	129.36
1 → 3	44.00
1 → 4	98.01
1 → 5	190.69
2 → 3	170.45
2 → 4	168.27
2 → 5	91.27
3 → 4	86.05
3 → 5	222.03
4 → 5	181.52

Также проинициализированы значения феромона на каждой дуге одним случайным значением: 0.876.

Для расчёта априорной эффективности из Формулы 1.1.1 использована Формула 1.3.2, чтобы иметь возможность проводить расчёты с точностью до пяти знаков после запятой.

$$\eta_{ij} = \frac{100}{d_{ij}} \tag{1.3.2}$$

1.3.1 Первая итерация

Для упрощения расчётов муравьи начинают свой путь с первой вершины.

Далее приведены расчёты вероятностей перехода первого муравья с 1-й вершины:

$$p_{12}^1 = \frac{0.876^1 * 0.494^2}{0.00403} = 0.005$$
$$p_{13}^1 = \frac{0.876^1 * 0.941^2}{0.00403} = 0.019$$
$$p_{14}^1 = \frac{0.876^1 * 0.431^2}{0.00403} = 0.004$$
$$p_{15}^1 = \frac{0.876^1 * 0.535^2}{0.00403} = 0.006$$
$$p_{16}^1 = \frac{0.876^1 * 6.667^2}{0.00403} = 0.965$$

Сгенерировано случайное число r = 0.783, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.005 < r = 0.783$$
$$P_3 = 0.025 < r = 0.783$$
$$P_4 = 0.029 < r = 0.783$$
$$P_5 = 0.035 < r = 0.783$$
$$P_6 = 1.000 \geq r = 0.783$$

Происходит переход на вершину 6. Далее приведены расчёты вероятностей перехода первого муравья с 6-й вершины:

$$p_{62}^1 = \frac{0.876^1 * 0.524^2}{0.00017} = 0.139$$
$$p_{63}^1 = \frac{0.876^1 * 1.096^2}{0.00017} = 0.606$$
$$p_{64}^1 = \frac{0.876^1 * 0.450^2}{0.00017} = 0.102$$
$$p_{65}^1 = \frac{0.876^1 * 0.551^2}{0.00017} = 0.153$$

Сгенерировано случайное число r = 0.044, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.139 \geq r = 0.044$$

Происходит переход на вершину 2. Далее приведены расчёты вероятностей перехода первого муравья с 2-й вершины:

$$p_{23}^1 = \frac{0.876^1 * 0.773^2}{0.00060} = 0.088$$
$$p_{24}^1 = \frac{0.876^1 * 2.273^2}{0.00060} = 0.759$$
$$p_{25}^1 = \frac{0.876^1 * 1.020^2}{0.00060} = 0.153$$

Сгенерировано случайное число r = 0.674, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_3 = 0.088 < r = 0.674$$
$$P_4 = 0.847 \geq r = 0.674$$

Происходит переход на вершину 4. Далее приведены расчёты вероятностей перехода первого муравья с 4-й вершины:

$$p_{43}^1 = \frac{0.876^1 * 0.587^2}{0.00015} = 0.203$$
$$p_{45}^1 = \frac{0.876^1 * 1.162^2}{0.00015} = 0.797$$

Сгенерировано случайное число r = 0.994, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_3 = 0.203 < r = 0.994$$
$$P_5 = 1.000 \geq r = 0.994$$

Происходит переход на вершину 5. Далее приведены расчёты вероятностей перехода первого муравья с 5-й вершины:

$$p_{53}^1 = \frac{0.876^1 * 0.594^2}{0.00003} = 1.000$$

Сгенерировано случайное число r = 0.545, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_3 = 1.000 \geq r = 0.545$$

Происходит переход на вершину 3. Затем муравей возвращается на начальную вершину, и его путь завершается. Таким образом, путь первого муравья выглядит следующим образом: $0 \to 5 \to 1 \to 3 \to 4 \to 2 \to 0$. Длина пути равна: $15.0 + 190.69 + 44.0 + 86.05 + 168.27 + 106.23 = 610.236$.

Далее приведены расчёты вероятностей перехода второго муравья с 1-й вершины:

$$p_{12}^2 = \frac{0.876^1 * 0.494^2}{0.00403} = 0.005$$
$$p_{13}^2 = \frac{0.876^1 * 0.941^2}{0.00403} = 0.019$$
$$p_{14}^2 = \frac{0.876^1 * 0.431^2}{0.00403} = 0.004$$

$$p_{15}^2 = \frac{0.876^1 * 0.535^2}{0.00403} = 0.006$$
$$p_{16}^2 = \frac{0.876^1 * 6.667^2}{0.00403} = 0.965$$

Сгенерировано случайное число r = 0.145, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.005 < r = 0.145$$
$$P_3 = 0.025 < r = 0.145$$
$$P_4 = 0.029 < r = 0.145$$
$$P_5 = 0.035 < r = 0.145$$
$$P_6 = 1.000 \geq r = 0.145$$

Происходит переход на вершину 6. Далее приведены расчёты вероятностей перехода второго муравья с 6-й вершины:

$$p_{62}^2 = \frac{0.876^1 * 0.524^2}{0.00017} = 0.139$$
$$p_{63}^2 = \frac{0.876^1 * 1.096^2}{0.00017} = 0.606$$
$$p_{64}^2 = \frac{0.876^1 * 0.450^2}{0.00017} = 0.102$$
$$p_{65}^2 = \frac{0.876^1 * 0.551^2}{0.00017} = 0.153$$

Сгенерировано случайное число r = 0.071, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.139 \geq r = 0.071$$

Происходит переход на вершину 2. Далее приведены расчёты вероятностей перехода второго муравья с 2-й вершины:

$$p_{23}^2 = \frac{0.876^1 * 0.773^2}{0.00060} = 0.088$$
$$p_{24}^2 = \frac{0.876^1 * 2.273^2}{0.00060} = 0.759$$
$$p_{25}^2 = \frac{0.876^1 * 1.020^2}{0.00060} = 0.153$$

Сгенерировано случайное число r = 0.919, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_3 = 0.088 < r = 0.919$$
$$P_4 = 0.847 < r = 0.919$$
$$P_5 = 1.000 \geq r = 0.919$$

Происходит переход на вершину 5. Далее приведены расчёты вероятностей перехода второго муравья с 5-й вершины:

$$p_{53}^2 = \frac{0.876^1 * 0.594^2}{0.00015} = 0.207$$
$$p_{54}^2 = \frac{0.876^1 * 1.162^2}{0.00015} = 0.793$$

Сгенерировано случайное число r = 0.548, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_3 = 0.207 < r = 0.548$$
$$P_4 = 1.000 \geq r = 0.548$$

Происходит переход на вершину 4. Далее приведены расчёты вероятностей перехода второго муравья с 4-й вершины:

$$p_{43}^{2} = \frac{0.876^{1} * 0.587^{2}}{0.00003} = 1.000$$

Сгенерировано случайное число r = 0.265, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_3 = 1.000 \geq r = 0.265$$

Происходит переход на вершину 3. Затем муравей возвращается на начальную вершину, и его путь завершается. Таким образом, путь второго муравья выглядит следующим образом: $0 \to 5 \to 1 \to 4 \to 3 \to 2 \to 0$. Длина пути равна: $15.0 + 190.69 + 98.01 + 86.05 + 170.45 + 106.23 = 666.423$.

Затем выполняется испарение феромона по Формуле 1.1.2:

$$\tau_{12} = (1 - 0.5) * 0.876 = 0.438$$
$$\tau_{13} = (1 - 0.5) * 0.876 = 0.438$$
$$\tau_{14} = (1 - 0.5) * 0.876 = 0.438$$
$$\tau_{15} = (1 - 0.5) * 0.876 = 0.438$$
$$\dots$$
$$\tau_{65} = (1 - 0.5) * 0.876 = 0.438$$

Изменение концентрации феромона происходит по Формуле 1.1.3, где значение Q принято за 100. Далее отображены только значения феромонов, которые после данной итерации изменились на ненулевую величину.

$$\tau_{16} = 0.438 + 0.314 = 0.752$$
$$\tau_{24} = 0.438 + 0.164 = 0.602$$
$$\tau_{25} = 0.438 + 0.150 = 0.588$$

$$\tau_{31} = 0.438 + 0.314 = 0.752$$
$$\tau_{43} = 0.438 + 0.150 = 0.588$$
$$\tau_{45} = 0.438 + 0.164 = 0.602$$
$$\tau_{53} = 0.438 + 0.164 = 0.602$$
$$\tau_{54} = 0.438 + 0.150 = 0.588$$
$$\tau_{62} = 0.438 + 0.314 = 0.752$$

1.3.2 Вторая итерация

Далее приведены расчёты вероятностей перехода первого муравья с 1-й вершины:

$$p_{12}^{1} = \frac{0.438^{1} * 0.494^{2}}{0.00341} = 0.003$$
$$p_{13}^{1} = \frac{0.438^{1} * 0.941^{2}}{0.00341} = 0.011$$
$$p_{14}^{1} = \frac{0.438^{1} * 0.431^{2}}{0.00341} = 0.002$$
$$p_{15}^{1} = \frac{0.438^{1} * 0.535^{2}}{0.00341} = 0.004$$
$$p_{16}^{1} = \frac{0.752^{1} * 6.667^{2}}{0.00341} = 0.979$$

Сгенерировано случайное число r = 0.954, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.003 < r = 0.954$$
$$P_3 = 0.015 < r = 0.954$$
$$P_4 = 0.017 < r = 0.954$$
$$P_5 = 0.021 < r = 0.954$$
$$P_6 = 1.000 \geq r = 0.954$$

Происходит переход на вершину 6. Далее приведены расчёты вероятностей перехода первого муравья с 6-й вершины:

$$p_{62}^{1} = \frac{0.752^{1} * 0.524^{2}}{0.00010} = 0.217$$
$$p_{63}^{1} = \frac{0.438^{1} * 1.096^{2}}{0.00010} = 0.551$$
$$p_{64}^{1} = \frac{0.438^{1} * 0.450^{2}}{0.00010} = 0.093$$
$$p_{65}^{1} = \frac{0.438^{1} * 0.551^{2}}{0.00010} = 0.139$$

Сгенерировано случайное число r = 0.098, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.217 \geq r = 0.098$$

Происходит переход на вершину 2. Далее приведены расчёты вероятностей перехода первого муравья с 2-й вершины:

$$p_{23}^{1} = \frac{0.438^{1} * 0.773^{2}}{0.00040} = 0.066$$
$$p_{24}^{1} = \frac{0.602^{1} * 2.273^{2}}{0.00040} = 0.781$$
$$p_{25}^{1} = \frac{0.588^{1} * 1.020^{2}}{0.00040} = 0.154$$

Сгенерировано случайное число r = 0.065, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_3 = 0.066 \geq r = 0.065$$

Происходит переход на вершину 3. Далее приведены расчёты вероятностей перехода первого муравья с 3-й вершины:

$$p_{34}^{1} = \frac{0.438^{1} * 0.587^{2}}{0.00003} = 0.494$$
$$p_{35}^{1} = \frac{0.438^{1} * 0.594^{2}}{0.00003} = 0.506$$

Сгенерировано случайное число r = 0.467, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_4 = 0.494 \geq r = 0.467$$

Происходит переход на вершину 4. Далее приведены расчёты вероятностей перехода первого муравья с 4-й вершины:

$$p_{45}^{1} = \frac{0.602^{1} * 1.162^{2}}{0.00008} = 1.000$$

Сгенерировано случайное число r = 0.958, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_5 = 1.000 \geq r = 0.958$$

Происходит переход на вершину 5. Затем муравей возвращается на начальную вершину, и его путь завершается. Таким образом, путь первого муравья выглядит следующим образом: $0 \to 5 \to 1 \to 2 \to 3 \to 4 \to 0$. Длина пути равна: $15.0 + 190.69 + 129.36 + 170.45 + 86.05 + 186.85 = 778.394$.

Далее приведены расчёты вероятностей перехода второго муравья с 1-й вершины:

$$p_{12}^2 = \frac{0.438^1 * 0.494^2}{0.00341} = 0.003$$
$$p_{13}^2 = \frac{0.438^1 * 0.941^2}{0.00341} = 0.011$$
$$p_{14}^2 = \frac{0.438^1 * 0.431^2}{0.00341} = 0.002$$
$$p_{15}^2 = \frac{0.438^1 * 0.535^2}{0.00341} = 0.004$$
$$p_{16}^2 = \frac{0.752^1 * 6.667^2}{0.00341} = 0.979$$

Сгенерировано случайное число r = 0.866, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.003 < r = 0.866$$
$$P_3 = 0.015 < r = 0.866$$
$$P_4 = 0.017 < r = 0.866$$
$$P_5 = 0.021 < r = 0.866$$
$$P_6 = 1.000 \geq r = 0.866$$

Происходит переход на вершину 6. Далее приведены расчёты вероятностей перехода второго муравья с 6-й вершины:

$$p_{62}^2 = \frac{0.752^1 * 0.524^2}{0.00010} = 0.217$$
$$p_{63}^2 = \frac{0.438^1 * 1.096^2}{0.00010} = 0.551$$
$$p_{64}^2 = \frac{0.438^1 * 0.450^2}{0.00010} = 0.093$$

$$p_{65}^2 = \frac{0.438^1 * 0.551^2}{0.00010} = 0.139$$

Сгенерировано случайное число r = 0.879, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.217 < r = 0.879$$
$$P_3 = 0.768 < r = 0.879$$
$$P_4 = 0.861 < r = 0.879$$
$$P_5 = 1.000 \geq r = 0.879$$

Происходит переход на вершину 5. Далее приведены расчёты вероятностей перехода второго муравья с 5-й вершины:

$$p_{52}^2 = \frac{0.438^1 * 1.020^2}{0.00015} = 0.312$$
$$p_{53}^2 = \frac{0.602^1 * 0.594^2}{0.00015} = 0.145$$
$$p_{54}^2 = \frac{0.588^1 * 1.162^2}{0.00015} = 0.543$$

Сгенерировано случайное число r = 0.343, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.312 < r = 0.343$$
$$P_3 = 0.457 \geq r = 0.343$$

Происходит переход на вершину 3. Далее приведены расчёты вероятностей перехода второго муравья с 3-й вершины:

$$p_{32}^2 = \frac{0.438^1 * 0.773^2}{0.00004} = 0.635$$
$$p_{34}^2 = \frac{0.438^1 * 0.587^2}{0.00004} = 0.365$$

Сгенерировано случайное число r = 0.329, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_2 = 0.635 \geq r = 0.329$$

Происходит переход на вершину 2. Далее приведены расчёты вероятностей перехода второго муравья с 2-й вершины:

$$p_{24}^2 = \frac{0.602^1 * 2.273^2}{0.00031} = 1.000$$

Сгенерировано случайное число r = 0.841, которое определяет, на какую из вершин будет совершён переход. Для этого рассчитанные вероятности складываются в накопительную сумму.

$$P_4 = 1.000 \geq r = 0.841$$

Происходит переход на вершину 4. Затем муравей возвращается на начальную вершину, и его путь завершается. Таким образом, путь второго муравья выглядит следующим образом: $0 \to 5 \to 4 \to 2 \to 1 \to 3 \to 0$. Длина пути равна: $15.0 + 181.52 + 168.27 + 129.36 + 44.0 + 232.04 = 770.178$.

Затем выполняется испарение феромона по Формуле 1.1.2 аналогично предыдущей итерации.

Изменение концентрации феромона происходит по Формуле 1.1.3, где значение Q принято за 100. Далее отображены только значения феромонов, которые после данной итерации изменились на ненулевую величину.

$$\tau_{16} = 0.376 + 0.258 = 0.634$$
$$\tau_{23} = 0.219 + 0.128 = 0.347$$
$$\tau_{24} = 0.301 + 0.130 = 0.431$$
$$\tau_{32} = 0.219 + 0.130 = 0.349$$
$$\tau_{34} = 0.219 + 0.128 = 0.347$$
$$\tau_{41} = 0.219 + 0.130 = 0.349$$
$$\tau_{45} = 0.301 + 0.128 = 0.429$$
$$\tau_{51} = 0.219 + 0.128 = 0.347$$
$$\tau_{53} = 0.301 + 0.130 = 0.431$$
$$\tau_{62} = 0.376 + 0.128 = 0.504$$
$$\tau_{65} = 0.219 + 0.130 = 0.349$$

1.4 Программная реализация

Для реализации расчётов алгоритма муравьиной колонии написан программный код на языке Python 3.12.

В программной реализации зафиксированы следующие параметры:

- количество вершин в графе: 10;
- количество муравьёв: 10;
- количество итераций: 40;
- $\alpha = 1$;
- $\beta = 5$;
- $\rho = 0.5$;
- $Q = 100$.

Код реализации алгоритма муравьиной колонии представлен в Листинге А.1.

На Рисунке 1.4.1 представлен результат выполнения программы для нахождение минимального пути в графе — графики зависимости лучшего пути и среднего пути от номера итерации.

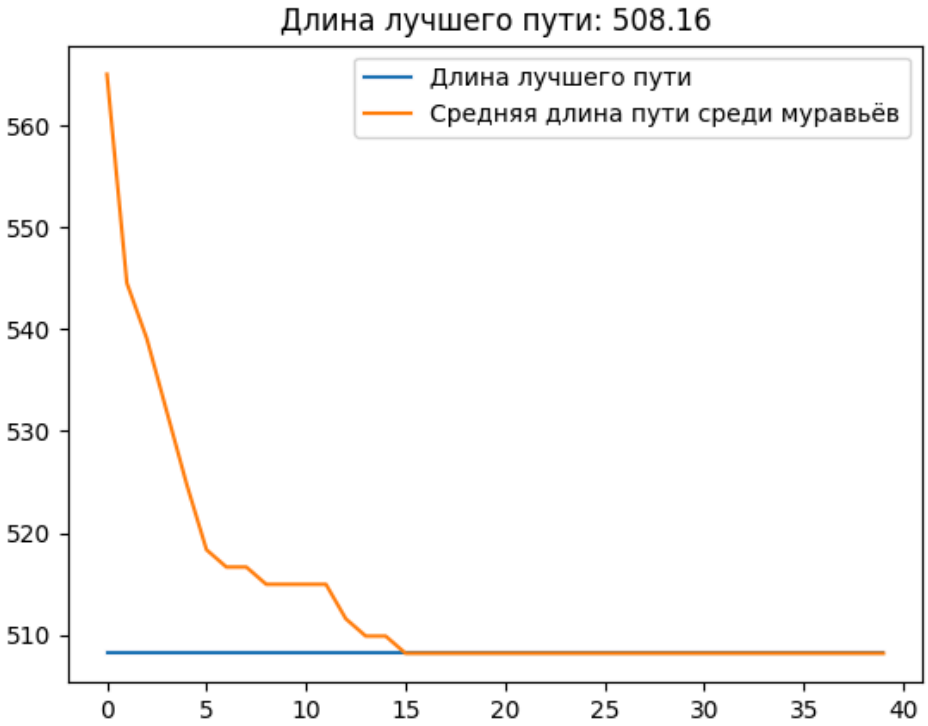

Рисунок 1.4.1 — График зависимости длины пути от номера итерации

На Рисунке 1.4.2 представлен лучший путь, построенный муравьями в результате выполнения алгоритма.

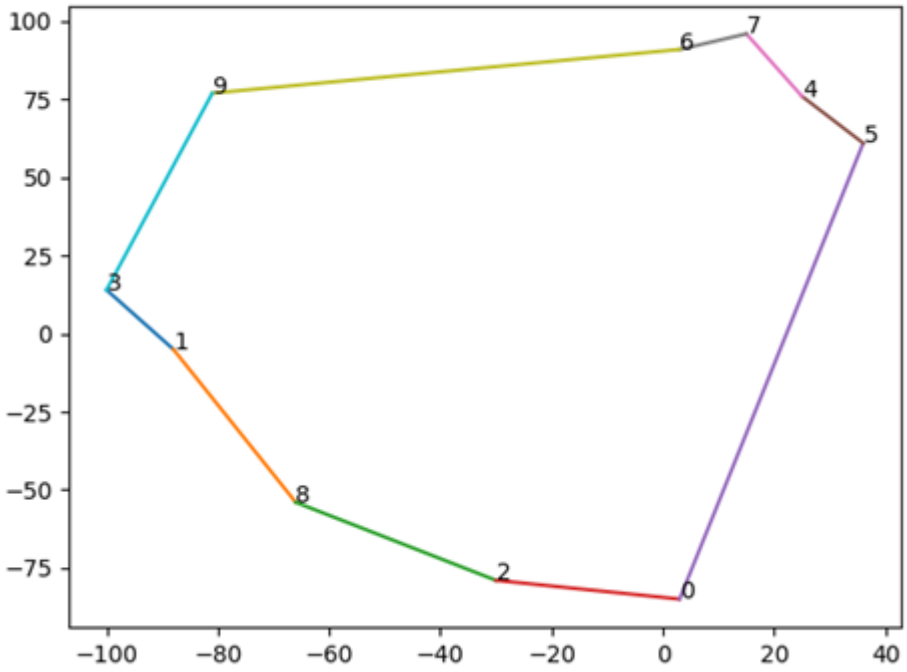

Рисунок 1.4.2 — Построенный путь

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы выполнены поставленные задачи — изучен муравьиный алгоритм, произведён его ручной расчёт для решения задачи коммивояжёра, а также разработаны программы на языке Python для нахождения оптимального пути в графе.

В заключение можно отметить, что муравьиный алгоритм является мощным инструментом для решения задач оптимизации, в которых стандартные методы недостаточно эффективны из-за наличия множества решений, среди которых нужно найти только одно оптимальное. Алгоритм отлично справляется с нахождением оптимальных путей в графе, однако его сходимость является не очень высокой, а также результативность алгоритма сильно зависит от настройки свободных параметров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Карпенко, А. П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учебное пособие / А. П. Карпенко — 3-е изд. — Москва: Издательство МГТУ им. Н. Э. Баумана, 2021. — 446 с.

2. Пряжников, В. Алгоритм имитации отжига [Электронный ресурс]. URL: https://pryazhnikov.com/notes/simulated-annealing/ (Дата обращения: 12.11.2024).

3. Сорокин, А. Б. Введение в роевой интеллект: теория, расчеты и приложения [Электронный ресурс]: Учебно-методическое пособие / А. Б. Сорокин — М.: Московский технологический университет (МИРЭА), 2019.

4. Rastrigin function [Электронный ресурс]: Википедия. URL: https://en.wikipedia.org/wiki/Rastrigin_function (Дата обращения: 01.11.2024).

5. Wang, Q., Zeng, J., Song, W. A New Electromagnetism-like Algorithm with Chaos Optimization 2010. С. 535–538.

ПРИЛОЖЕНИЯ

Приложение А — Реализация муравьиного алгоритма на языке Python.

Приложение А

Реализация муравьиного алгоритма на языке Python

Листинг А.1 — Реализация муравьиного алгоритма

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
import time
from math import sqrt


class GraphElement:
    def __init__(self, distance: float | None = None, pheromone: float = 0):
        self.distance = distance
        self.pheromone = pheromone


class Path:
    graph: list[list[GraphElement]] | None = None
    x: np.ndarray | None = None
    y: np.ndarray | None = None

    def __init__(self,
                 path: list[int] | None = None,
                 graph: list[list[GraphElement]] | None = None,
                 x: np.ndarray | None = None,
                 y: np.ndarray | None = None):
        if path is None:
            self.path = [0]
        else:
            self.path = path

        if Path.graph is None:
            Path.graph = graph
        if Path.x is None:
            Path.x = x
        if Path.y is None:
            Path.y = y

        if Path.graph is None:
            raise Exception('Граф не построен')

     def create_new_path(self, alpha: float, beta: float, start_position: int
= 0):
        self.path = [start_position]
        while not all([x in self.path for x in range(len(Path.graph))]):
            current_place = self.path[-1]
            possible_places = [
                i for i in range(len(Path.graph))
                if i not in self.path
            ]
            pheromones = []
            for next_place in possible_places:
             pheromones.append((Path.graph[current_place][next_place].pheromone
** alpha
                                                * (1 / Path.graph[current_place]
[next_place].distance) ** beta))
            total_pheromone = sum(pheromones)
            pheromones = [elem / total_pheromone for elem in pheromones]

            probability = np.random.random()
            total = 0
            for next_place, ph in zip(possible_places, pheromones):
                total += ph
                if probability <= total:
                    self.path.append(next_place)
                    break
```

Продолжение Листинга А.1

```
        self.path.append(self.path[0])

    @property
    def length(self) -> float:
        length = 0
        for i in range(1, len(self.path)):
            length += Path.graph[self.path[i]][self.path[i - 1]].distance
        return length

    def __str__(self) -> str:
        return ', '.join(map(str, self.path))

    def print_verbose(self, start_at_zero: bool = False):
        result_str = ''
        if start_at_zero:
            if self.path.index(0) != 0:
                index = self.path.index(0)
                first_part = self.path[index:]
                second_part = self.path[1:index]
                path = first_part + second_part + [0]
            else:
                path = self.path
        else:
            path = self.path
        for i, point in enumerate(path):
            result_str += str(point)

            if i != len(path) - 1:
                result_str += ' -> '
        return result_str

    def print_length(self):
        cum_length = []
        for i in range(1, len(self.path)):
            edge_len = Path.graph[self.path[i]][self.path[i - 1]].distance
            if not cum_length:
                cum_length.append(edge_len)
            else:
                cum_length.append(edge_len)
        return " + ".join(map(lambda x: str(round(x, 2)), cum_length))

    def draw_path(self):
        for i in range(len(Path.graph)):
            plt.text(Path.x[i], Path.y[i], f'{i}')
        for v0, v1 in zip(self.path, self.path[1:]):
            x = (Path.x[v0], Path.x[v1])
            y = (Path.y[v0], Path.y[v1])
            plt.plot(x, y)
        plt.show()


class Ant:
    def __init__(self):
        self.path = Path()


class AntColony:
    def __init__(self, ant_count: int = 10):
        self.ants = [Path() for _ in range(ant_count)]
        self.vapor_rate = 0.95
        self.alpha = 6
        self.beta = 7

    def solution_step(self):
        for i, ant in enumerate(self.ants):
            ant.create_new_path(self.alpha, self.beta, i)
            # print(ant.print_verbose())

        for i in range(len(Path.graph)):
            for j in range(len(Path.graph)):
                if i != j:
```

Продолжение Листинга А.1

```
                    Path.graph[i][j].pheromone *= (1 - self.vapor_rate)

           delta_pheromones = [[0 for _ in range(len(Path.graph))] for _ in
range(len(Path.graph))]
        for ant in self.ants:
            delta_pheromone = 100 / ant.length
            for i, j in zip(ant.path, ant.path[1:]):
                delta_pheromones[i][j] += delta_pheromone

        for i in range(len(Path.graph)):
            for j in range(len(Path.graph)):
                Path.graph[i][j].pheromone += delta_pheromones[i][j]


        best_path = self.ants[0]
        best_length = best_path.length
        for ant in self.ants[1:]:
            if ant.length < best_length:
                best_path = ant
                best_length = best_path.length

        return Path(best_path.path)


class Solution:

    def __init__(self, file_path):
        self.file_path = file_path
        self.best_path: Path | None = None

    def create_graph(self):
        df = pd.read_csv(self.file_path)
        x = df['x'].to_numpy()
        y = df['y'].to_numpy()
         self.graph = graph = [[GraphElement() for _ in range(len(df))] for _
in range(len(df))]
        pheromone = 0.78498
        for i in range(len(df)):
            for j in range(i, len(df)):
                    graph[i][j].distance = graph[j][i].distance = sqrt((x[i] -
x[j])**2 + (y[i] - y[j])**2)
                if i != j:
                    graph[i][j].pheromone = graph[j][i].pheromone = pheromone
        Path(graph=graph, x=x, y=y)

    def solve(self):
        ant_colony = AntColony(ant_count=6)
        steps = 40
        history = []
        mean_history = []
        for step in range(steps):
            print(f'Итерация {step + 1} / {steps}')
            if (step + 1) % (steps // 5) == 0 or step == 0:
                for ant in ant_colony.ants:
                    print(ant.print_verbose(start_at_zero=True))
                    print(ant.length)
            current_path = ant_colony.solution_step()
            mean = 0
            for ant in ant_colony.ants:
                mean += ant.length
            mean /= len(ant_colony.ants)
            mean_history.append(mean)
                        if self.best_path is None or current_path.length <
self.best_path.length:
                self.best_path = current_path
            history.append(self.best_path.length)
                                                        print(f'Лучший    путь:
{self.best_path.print_verbose(start_at_zero=True)}')
            print(f'Длина лучшего пути: {self.best_path.length}')
            print('\n\n\n')
        fig, ax = plt.subplots()
```

Окончание Листинга А.1

```
        ax.set_title(f'Длина лучшего пути: {self.best_path.length:.2f}')
        plt.plot(history, label='Длина лучшего пути')
        plt.plot(mean_history, label='Средняя длина пути среди муравьёв')
        plt.legend(loc='upper right')
        plt.show()
        self.best_path.draw_path()



def main():
    solution = Solution('../prac2/backup_10.csv')
    solution.create_graph()
    solution.solve()


if __name__ == '__main__':
    main()
```