



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе № 6
по курсу «Анализ алгоритмов»
на тему: «Задача коммивояжера»

Студент ИУ7-54Б
(Группа)

(Подпись, дата)

Булдаков М.
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Волкова Л. Л.
(И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

| | |
|---|-----------|
| ВВЕДЕНИЕ | 3 |
| 1 Аналитический раздел | 4 |
| 1.1 Описание задачи | 4 |
| 1.2 Алгоритмы решения задачи | 4 |
| 2 Конструкторский раздел | 7 |
| 2.1 Требования к программному обеспечению | 7 |
| 2.2 Описание используемых типов данных | 7 |
| 2.3 Разработка алгоритмов | 8 |
| 2.4 Оценка трудоемкости алгоритмов | 13 |
| 2.4.1 Трудоемкость алгоритма полного перебора | 13 |
| 2.4.2 Трудоемкость муравьиного алгоритма | 14 |
| 3 Технологический раздел | 15 |
| 3.1 Средства реализации | 15 |
| 3.2 Сведения о модулях программы | 15 |
| 3.3 Реализация алгоритмов | 15 |
| 4 Исследовательский раздел | 22 |
| 4.1 Демонстрация работы программы | 22 |
| 4.2 Технические характеристики | 24 |
| 4.3 Время выполнения реализаций алгоритмов | 24 |
| 4.4 Параметризация муравьиного алгоритма | 27 |
| 4.4.1 Класс данных | 27 |
| ЗАКЛЮЧЕНИЕ | 30 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 31 |
| ПРИЛОЖЕНИЕ А | 32 |

ВВЕДЕНИЕ

Задача коммивояжера является одной из классических задач комбинаторной оптимизации, привлекающей внимание исследователей и практиков в области логистики, транспорта и информационных технологий. В контексте логистики, эта задача применяется для оптимизации маршрутов доставки товаров и грузов, что позволяет сократить затраты на перевозку, уменьшить время доставки и улучшить эффективность работы логистических компаний [1].

Цель данной лабораторной работы — рассмотреть алгоритмы решения задачи коммивояжера в случае построения карты перемещений для воздухоплавателей.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- описать алгоритмы решения задачи коммивояжера;
- спроектировать программное обеспечение, реализующее алгоритмы решения задачи коммивояжера;
- выбрать инструменты для реализации и замера процессорного времени выполнения реализаций решения задачи;
- проанализировать затраты реализаций алгоритмов по времени.

1 Аналитический раздел

В данном разделе будут описаны алгоритмы решения задачи коммивояжера.

1.1 Описание задачи

Задача коммивояжера — одна из самых известных задач комбинаторной оптимизации, заключающаяся в поиске наиболее выгодного маршрута, проходящего через указанные города и возвращающегося в начальный пункт.

В общем случае задача формулируется следующим образом: имеется N городов, для каждой пары которых известно расстояние между ними. Требуется найти такой маршрут, проходящий через каждый город по одному разу (и возвращающийся в исходный город), при этом сумма всех расстояний на этом маршруте должна быть наименьшей из возможных [1].

В данной работе будет рассматриваться вариант задачи с незамкнутым маршрутом, т. е. без одного последнего перехода. Стоимость пути между городами будет отличаться в различных направлениях, поэтому граф будет ориентированным.

1.2 Алгоритмы решения задачи

Алгоритм полного перебора

Задача может быть решена перебором всех вариантов объезда и выбором оптимального. Очевидно, что при полном переборе будет найден самый кратчайший маршрут, но при этом для перебора необходимо будет выполнить порядка $O(N!)$ операций, где N — количество городов, что является тяжелой задачей даже для современных ЭВМ при N порядка сотни.

Муравьиный алгоритм (без элитных муравьев)

Муравьиный алгоритм — алгоритм решения задачи коммивояжера, основанный на принципе поведения колонии муравьев [2].

Муравьи действуют, руководствуясь органами чувств. Каждый муравей оставляет на своем пути феромоны, чтобы другие могли ориентироваться. При большом количестве муравьев наибольшее количество феромона остается на наиболее посещаемом пути, посещаемость же может быть связана с длинами ребер. Муравьи используют не прямой обмен информацией через окружающую

среду посредством феромона.

Основная идея заключается в том, что выделяются две фазы: день и ночь. В фазу дня каждый муравей k строит один маршрут, вечером обновляется лучшая траектория. В фазу ночи обновляется матрица феромона.

Каждый муравей имеет 3 способности:

- 1) Зрение — муравей k , стоя в городе i , может оценить привлекательность ребра $i - j$;
- 2) Обоняние — муравей чувствует концентрацию феромона $\tau_{ij}(t)$ на ребре $i-j$ в текущий день t ;
- 3) Память — муравей запоминает список, посещенных за текущий день t городов — $J_k(t)$.

Привлекательность ребра $i-j$ оценивается по формуле (1.1).

$$\eta_{ij} = \frac{1}{D_{ij}}, \quad (1.1)$$

где D_{ij} — метка ребра, D — матрица смежности.

Стоя в городе i , муравей k выбирает следующий город на основе вероятностного правила (1.2).

$$p_{ij,k} = \begin{cases} 0, j \in J_k, \\ \frac{\eta_{ij}^\alpha \cdot \tau_{ij}^\beta(t)}{\sum_{q \in J_k} \eta_{iq}^\alpha \cdot \tau_{iq}^\beta(t)}, \text{ иначе,} \end{cases} \quad (1.2)$$

где α — коэффициент жадности, β — коэффициент стадности, причем $\alpha + \beta = 1$.

После завершения движения всех муравьев (ночью, перед наступлением следующего дня), феромон обновляется по формуле (1.3).

$$\tau_{ij}(t+1) = \tau_{ij}(t) \cdot (1 - \rho) + \Delta\tau_{ij}(t), \quad (1.3)$$

где $\rho \in [0, 1]$ — коэффициент испарения феромона.

$$\Delta\tau_{ij}(t) = \sum_{k=1}^N \Delta\tau_{ij,k}(t). \quad (1.4)$$

$$\Delta\tau_{ij}^k(t) = \begin{cases} 0, \text{ муравей } k \text{ в день } t \text{ не ходил по ребру } i-j, \\ Q/L_k, \text{ иначе,} \end{cases} \quad (1.5)$$

где Q — квота феромона одного муравья на день, Q выбирается соразмерной длине лучшего маршрута в графе.

Чтобы значение феромона не обнулилось и не повлекло обнуление вероятности перехода по ребру, после расчета значения $\tau_{ij}(t+1)$ необходимо выполнять проверку матрицы феромона и все значения, которые меньше заданного порога, заменить на порог.

В данном алгоритме отсутствует полный перебор, что означает меньшую трудоемкость, чем для алгоритма полного перебора, но при этом лучшее решение не гарантируется.

Вывод

В данном разделе были описаны алгоритм полного перебора и муравьиный алгоритм для решения задачи коммивояжера.

2 Конструкторский раздел

В этом разделе будет представлено описание используемых типов данных, а также схематические изображения алгоритмов решения задачи коммивояжера.

2.1 Требования к программному обеспечению

Программа должна поддерживать два режима работы: режим массового замера времени и режим решения задачи коммивояжера.

Режим массового замера времени должен обладать следующей функциональностью:

- генерировать графы различного размера для проведения замеров;
- осуществлять массовый замер, используя сгенерированные данные;
- результаты массового замера должны быть представлены в виде таблицы и графика.

К режиму решения задачи коммивояжера выдвигается следующий ряд требований:

- возможность вводить матрицы смежности графов;
- наличие интерфейса для выбора действий;
- на выходе программы, стоимость и маршрут кратчайшей длины.

2.2 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие структуры и типы данных:

- граф — множество вершин и ребер между ними, задается с помощью матрицы смежности;
- матрица — двумерный массив чисел.

2.3 Разработка алгоритмов

На рисунке 2.1 представлена схема алгоритма решения задачи коммивояжера полным перебором. На рисунке 2.2 представлена схема муравьиного алгоритма. На рисунках 2.3 и 2.4 изображены схемы алгоритмов вспомогательных подпрограмм.

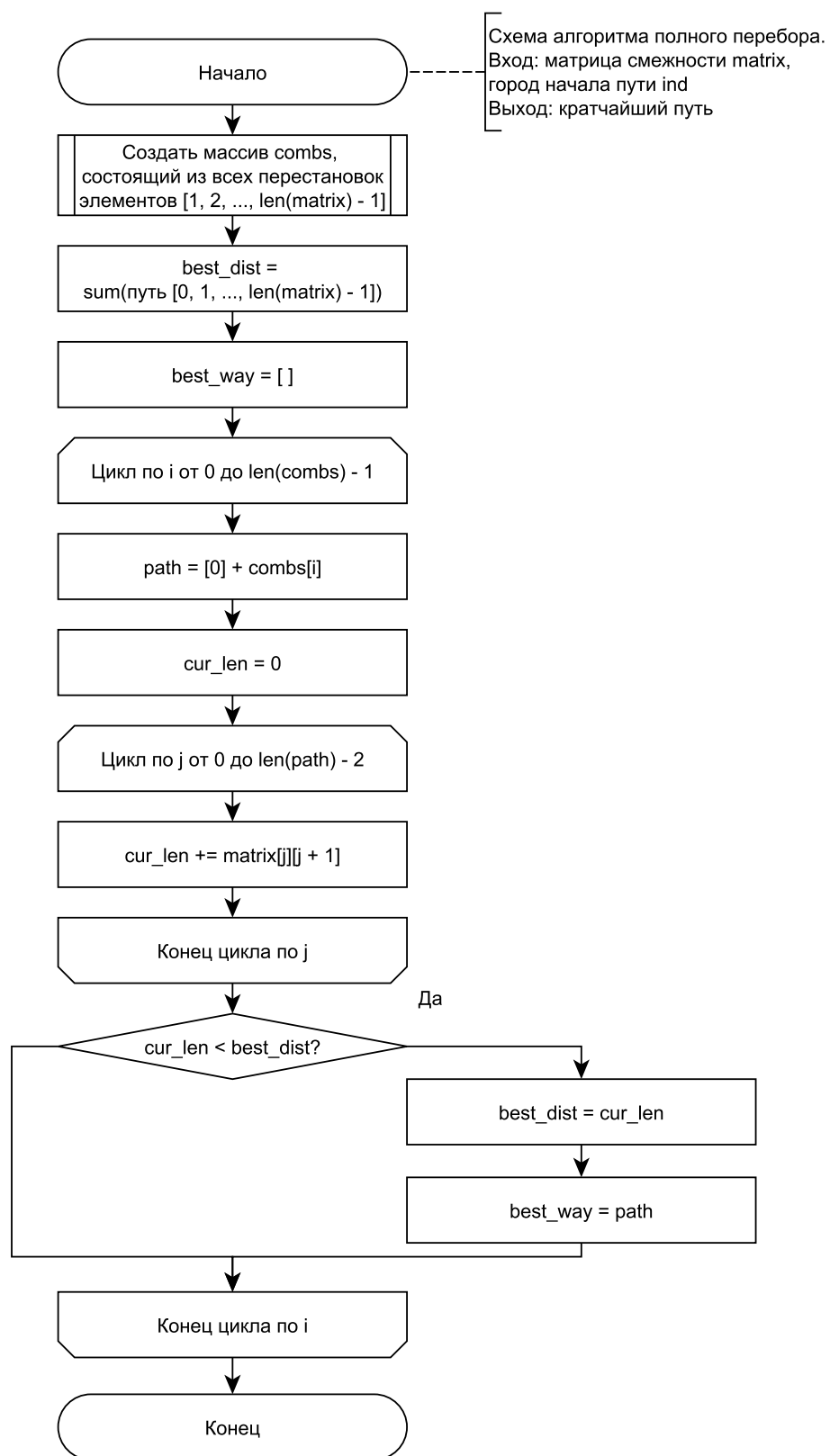


Рисунок 2.1 – Схема алгоритма решения задачи коммивояжера полным перебором

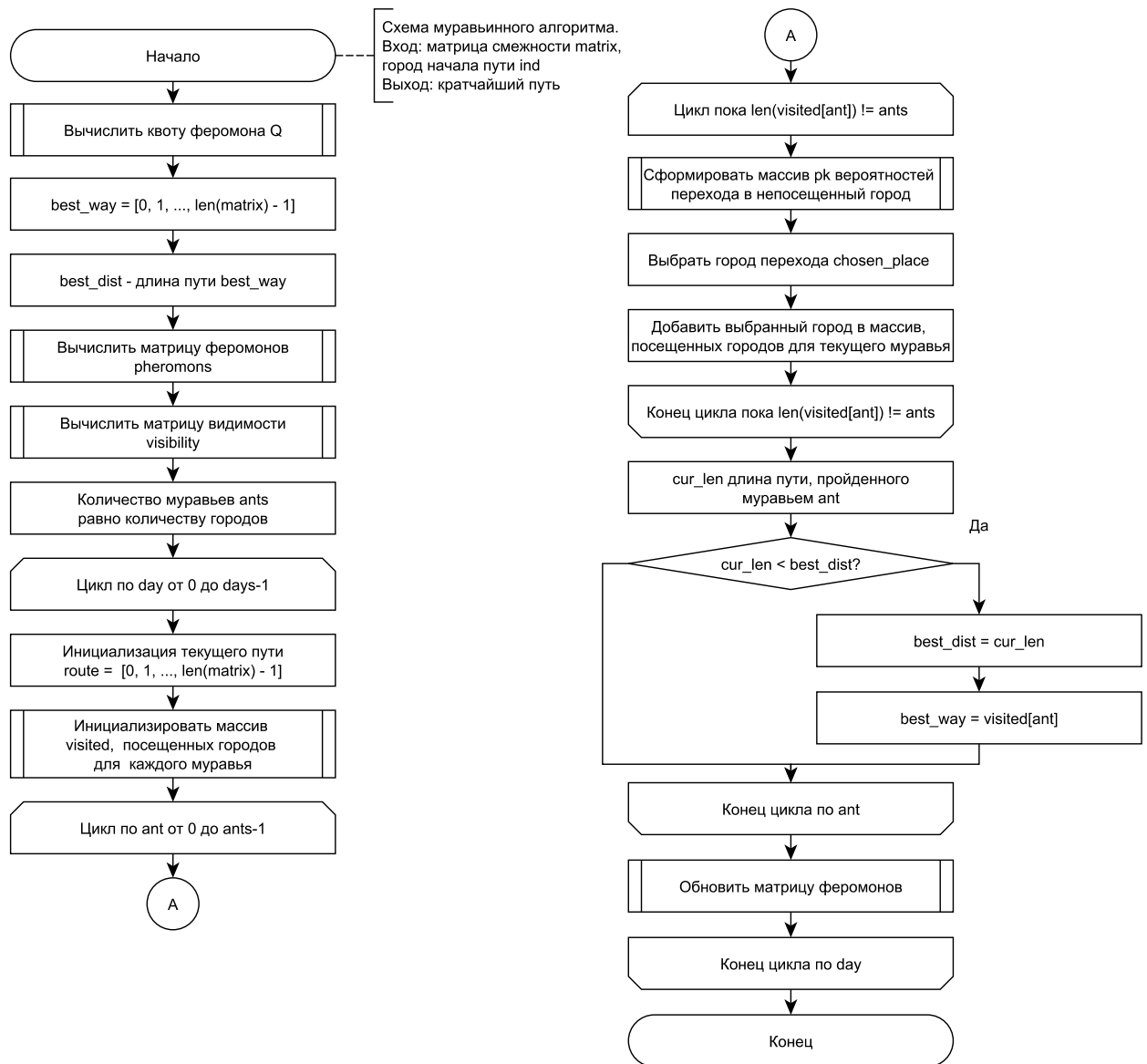


Рисунок 2.2 – Схема муравьиного алгоритма

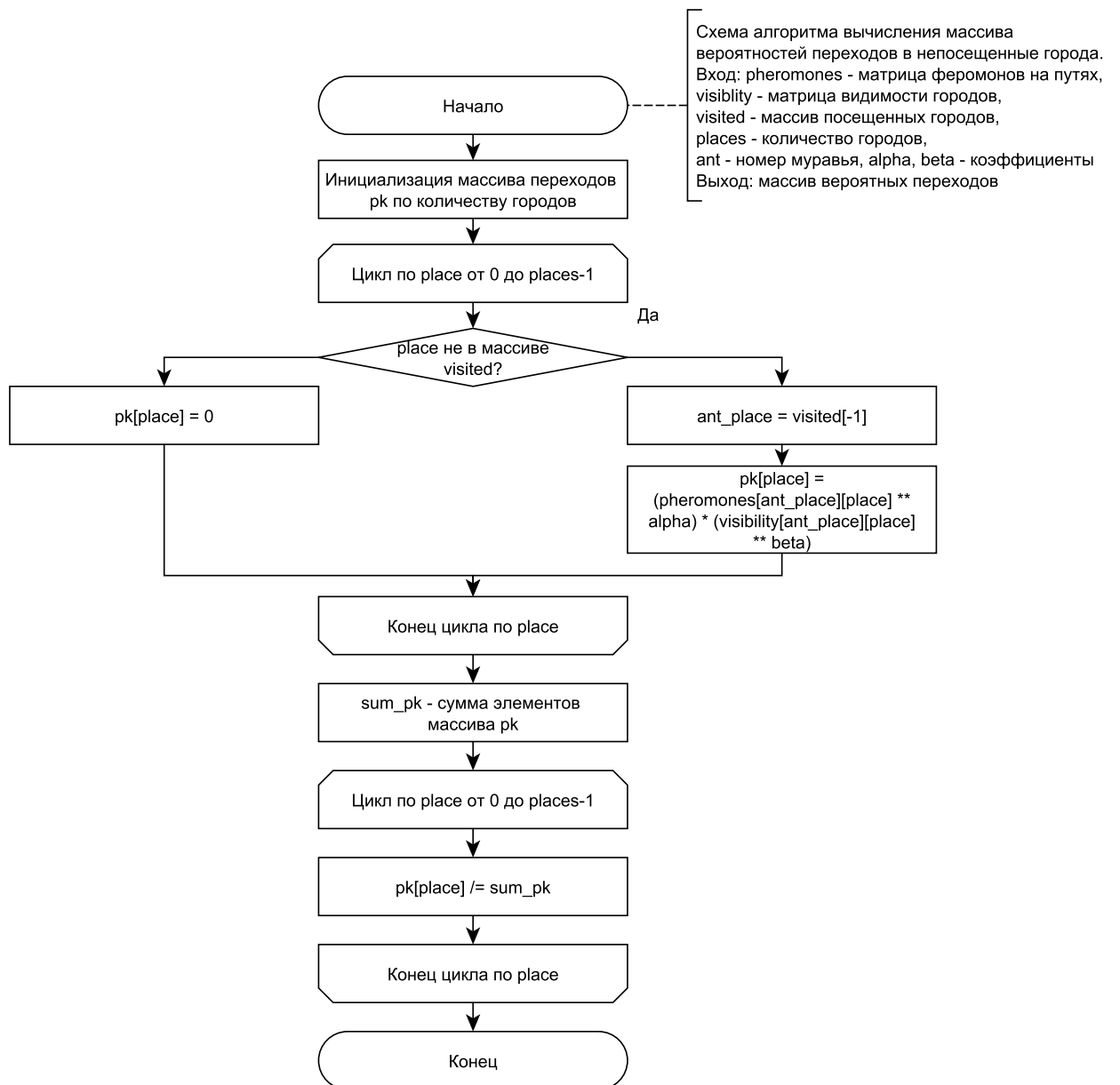


Рисунок 2.3 – Схема алгоритма вычисления массива вероятностей переходов в непосещенные города

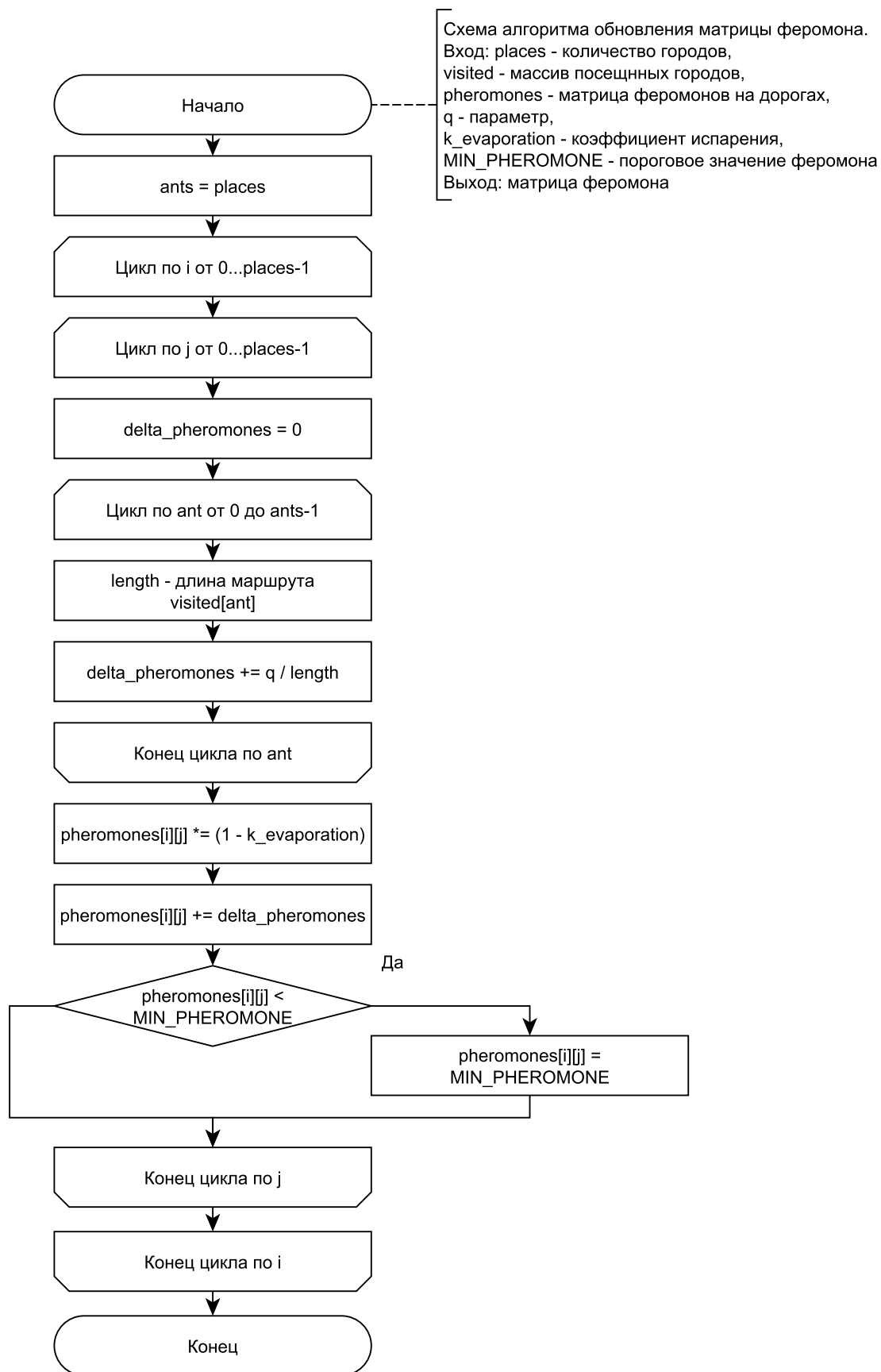


Рисунок 2.4 – Схема алгоритма обновления матрицы феромона

2.4 Оценка трудоемкости алгоритмов

Модель для оценки трудоемкости алгоритмов состоит из шести пунктов:

- 1) $+, -, =, + =, - =, ==, ||, \&\&, <, >, <=, >=, <<, >>, []$ — считается, что эти операции обладают трудоемкостью в 1 единицу;
- 2) $*, /, * =, / =, \%, **$ — считается, что эти операции обладают трудоемкостью в 2 единицы;
- 3) трудоемкость условного перехода принимается за 0;
- 4) трудоемкость условного оператора рассчитывается по формуле (2.1),

$$f_{if} = f_{\text{условия}} + \begin{cases} \min(f_1, f_2), & \text{лучший случай} \\ \max(f_1, f_2), & \text{худший случай} \end{cases}, \quad (2.1)$$

где f_1 — трудоемкость блока, который вычисляется при выполнении условия, а f_2 — трудоемкость блока, который вычисляется при невыполнении условия;

- 5) трудоемкость цикла рассчитывается по формуле (2.2),

$$f_{for} = f_{\text{инициализация}} + f_{\text{сравнения}} + M_{\text{итераций}} \cdot (f_{\text{тело}} + f_{\text{инкремент}} + f_{\text{сравнения}}); \quad (2.2)$$

- 6) вызов подпрограмм и передача параметров принимается за 0.

2.4.1 Трудоемкость алгоритма полного перебора

Добавление в конец списка считается операцией стоимостью в 1. Тогда трудоемкость алгоритма составления массива перестановок из n элементов оценивается по формуле (2.3).

$$f_{perm} = 9 \cdot n! + 3 \quad (2.3)$$

Трудоемкость расчета пути длины n считается как $6n - 4$. Тогда трудоемкость алгоритма полного перебора в худшем случае считается по формуле

(2.4).

$$f_{brut} = (6n + 11)(n - 1)! + 6n + 1 = O(n!) \quad (2.4)$$

2.4.2 Трудоемкость муравьиного алгоритма

Трудоемкость рассчитывается для n городов и T дней. Трудоемкость алгоритма вычисления массива вероятностей переходов в города, рассчитывается по формуле (2.5).

$$f_{pk} = 13n + 4 \quad (2.5)$$

Трудоемкость алгоритма обновления матрицы феромона, рассчитывается по формуле (2.6).

$$f_{update} = 6n^4 + n^3 + 19n^2 + 4n + 3 \quad (2.6)$$

Тогда трудоемкость муравьиного алгоритма рассчитывается по формуле (2.7).

$$f_{ant} = T(6n^4 + 15n^3 + 25n^2 + 15n + 7) + 11n^2 + 14n + 7 = O(Tn^4) \quad (2.7)$$

Вывод

На основе теоретических данных, полученных из аналитического раздела были построены схемы требуемых алгоритмов. Была введена модель оценки трудоемкости алгоритма, были рассчитаны трудоемкости алгоритмов в соответствии с этой моделью.

В результате теоретической оценки трудоемкостей алгоритмов выяснилось, что лучшей асимптотической оценкой обладает муравьиный алгоритм $O(Tn^4)$, где T — количество дней жизни колонии, а n — количество городов. Алгоритм полного перебора обладает асимптотической оценкой $O(n!)$.

3 Технологический раздел

В данном разделе будут приведены требования к программному обеспечению, средства реализации, листинг кода.

3.1 Средства реализации

Для реализации данной работы был выбран язык *Python* [3]. Данный выбор обусловлен следующим:

- язык поддерживает все структуры данных, которые выбраны в результате проектирования;
- язык позволяет реализовать все алгоритмы, выбранные в результате проектирования;
- язык позволяет замерять процессорное время с помощью модуля *time*.

Процессорное время было замерено с помощью функции *process_time()* из модуля *time* [4].

3.2 Сведения о модулях программы

Данная программа разбита на следующие модули:

- *main.py* — файл, содержащий функцию *main*;
- *algorithms.py* — файл, содержащий код реализаций алгоритмов решения задачи коммивояжера;
- *utils.py* — содержит вспомогательные функции работы с графами;
- *compare_time.py* — файл, в котором содержатся функции для замера и вывода времени выполнения реализаций алгоритмов.

3.3 Реализация алгоритмов

В листинге 3.1 приведена реализация алгоритма решения задачи полным перебором. В листинге 3.2 приведена реализация муравьиного алгоритма. В листингах 3.3 – 3.5 приведены реализации вспомогательных подпрограмм.

Листинг 3.1 – Функция решения задачи коммивояжера полным перебором

```
1 def brut(matrix, start_city):
2     size = len(matrix)
3     nodes = list(range(0, size))
4     nodes.pop(start_city)
5
6     min_dist = float("inf")
7     best_way = []
8
9     for comb in it.permutations(nodes):
10         comb = [start_city] + list(comb)
11         cur_dist = 0
12         for i in range(len(comb) - 1):
13             cur_dist += matrix[comb[i]][comb[i + 1]]
14
15         if cur_dist < min_dist:
16             min_dist = cur_dist
17             best_way = comb
18
19     return min_dist, best_way
```


Листинг 3.2 – Функция решения задачи коммивояжера муравьиным алгоритмом

```
1 def ants(matrix, alpha, beta, k_evaporation, days, start_city):
2     places = len(matrix)
3
4     q = get_q(matrix)
5     best_way = []
6     min_dist = float("inf")
7     pheromones = [[1 for i in range(places)] for j in
8         range(places)]
9     visibility = [
10         [(1.0 / matrix[i][j] if (i != j) else 0) for j in
11             range(len(matrix[i]))]
12         for i in range(len(matrix))
13     ]
14
15     ants = places
16     for day in range(days):
17         route = np.arange(places)
18         visited = [[start_city] for _ in range(ants)]
19         for ant in range(ants):
20             while len(visited[ant]) != ants:
21                 pk = find_ways(
22                     pheromones, visibility, visited, places,
23                     ant, alpha, beta
24                 )
25                 chosen_place = choose_place(pk)
26                 visited[ant].append(chosen_place - 1)
27
28                 cur_length = get_length(matrix, visited[ant])
29
30                 if cur_length < min_dist:
31                     min_dist = cur_length
32                     best_way = visited[ant]
33
34         pheromones = update_pheromones(
35             matrix, places, visited, pheromones, q, k_evaporation
36         )
37
38     return min_dist, best_way
```

Листинг 3.3 – Функция вычисления массива вероятностей переходов в непосещенные города

```
1 def find_ways(pheromones, visibility, visited, places, ant,
2   alpha, beta):
3
4     for place in range(places):
5         if place not in visited[ant]:
6             ant_place = visited[ant][-1]
7             pk[place] = pow(pheromones[ant_place][place], alpha)
8                 * pow(
9                     visibility[ant_place][place], beta
10                )
11         else:
12             pk[place] = 0
13
14     sum_pk = sum(pk)
15
16     for place in range(places):
17         pk[place] /= sum_pk
18
19     return pk
```

Листинг 3.4 – Функция обновляющая матрицу феромонов

```
1 MIN_PHEROMONE = 0.01
2
3
4 def update_pheromones(matrix, places, visited, pheromones, q,
5     k_evaporation):
6     ants = places
7
8     for i in range(places):
9         for j in range(places):
10             delta = 0
11             for ant in range(ants):
12                 length = get_length(matrix, visited[ant])
13                 delta += q / length
14
15             pheromones[i][j] *= 1 - k_evaporation
16             pheromones[i][j] += delta
17             if pheromones[i][j] < MIN_PHEROMONE:
18                 pheromones[i][j] = MIN_PHEROMONE
19
20     return pheromones
```

Листинг 3.5 – Вспомогательные функции

```
1 def choose_place(pk):
2     possibility = random()
3     choice = 0
4     chosen_place = 0
5     while (choice < possibility) and (chosen_place < len(pk)):
6         choice += pk[chosen_place]
7         chosen_place += 1
8
9     return chosen_place
10
11
12 def get_q(matrix):
13     q = 0
14     count = 0
15     for i in range(len(matrix)):
16         for j in range(len(matrix[i])):
17             if i != j:
18                 q += matrix[i][j]
19                 count += 1
20     return q / count
21
22
23 def get_length(matrix, route):
24     length = 0
25
26     for way_len in range(1, len(route)):
27         length += matrix[route[way_len - 1]][route[way_len]]
28
29     return length
```

Вывод

В данном разделе были разработаны спроектированные алгоритмы решения задачи коммивояжера.

4 Исследовательский раздел

В данном разделе будут приведены: пример работы программы, постановка эксперимента и сравнительный анализ алгоритмов на основе полученных данных.

4.1 Демонстрация работы программы

На рисунке 4.1 представлена демонстрация работы разработанного программного обеспечения, а именно показаны результаты решения задачи ком-

мивояжера для графа, заданного матрицей смежности

$$\begin{pmatrix} 0 & 6 & 1 & 9 & 9 \\ 1 & 0 & 6 & 2 & 7 \\ 1 & 4 & 0 & 6 & 5 \\ 8 & 5 & 5 & 0 & 8 \\ 5 & 1 & 4 & 8 & 0 \end{pmatrix}.$$

- 1 - Решить задачу для введенной матрицы полным перебором.
- 2 - Решить задачу для введенной матрицы муравьиным алгоритмом.
- 3 - Выполнить решение заготовленных матриц.
- 4 - Провести замерный эксперимент.
- 0 - Выйти.

Выберите пункт меню: 1

Введит количество городов: 5

Вводите матрицу

0 6 1 9 9

1 0 6 2 7

1 4 0 6 5

8 5 5 0 8

5 1 4 8 0

Введите начальный город: 0

Кратчайший путь [0, 2, 4, 1, 3] длиной 9

- 1 - Решить задачу для введенной матрицы полным перебором.
- 2 - Решить задачу для введенной матрицы муравьиным алгоритмом.
- 3 - Выполнить решение заготовленных матриц.
- 4 - Провести замерный эксперимент.
- 0 - Выйти.

Выберите пункт меню: 2

Введит количество городов: 5

Вводите матрицу

0 6 1 9 9

1 0 6 2 7

1 4 0 6 5

8 5 5 0 8

5 1 4 8 0

Введите начальный город: 0

Введите количество дней: 10

Кратчайший путь [0, 2, 4, 1, 3] длиной 9

Рисунок 4.1 – Демонстрация работы программы при решении задачи коммивояжера

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялись замеры по времени, следующие:

- процессор: AMD Ryzen 5 4600H 3 ГГц [5];
- оперативная память: 16 ГБайт;
- операционная система: Windows 10 Pro 64-разрядная система версии 22H2 [6].

При замерах времени ноутбук был включен в сеть электропитания и был нагружен только системными приложениями.

4.3 Время выполнения реализаций алгоритмов

Результаты замеров времени выполнения реализаций алгоритмов решения задачи коммивояжера приведены в таблице 4.1. Замеры времени проводились на полносвязных графах одного размера и усреднялись для каждого набора одинаковых экспериментов. Для реализации муравьиного алгоритма количество дней равно 10.

Таблица 4.1 – Время работы реализации алгоритмов решения задачи коммивояжера (в с)

| Количество городов | Полный перебор | Муравьиный |
|--------------------|-----------------------|-----------------------|
| 4 | $1.563 \cdot 10^{-5}$ | $1.422 \cdot 10^{-3}$ |
| 5 | $4.688 \cdot 10^{-5}$ | $3.125 \cdot 10^{-3}$ |
| 6 | $2.344 \cdot 10^{-4}$ | $5.750 \cdot 10^{-3}$ |
| 7 | $1.594 \cdot 10^{-3}$ | $1.005 \cdot 10^{-2}$ |
| 8 | $1.231 \cdot 10^{-2}$ | $1.664 \cdot 10^{-2}$ |
| 9 | $1.105 \cdot 10^{-1}$ | $2.506 \cdot 10^{-2}$ |

На рисунках 4.2 и 4.3 изображены графики зависимостей времени выполнения реализаций алгоритмов решения задачи коммивояжера от количества городов.

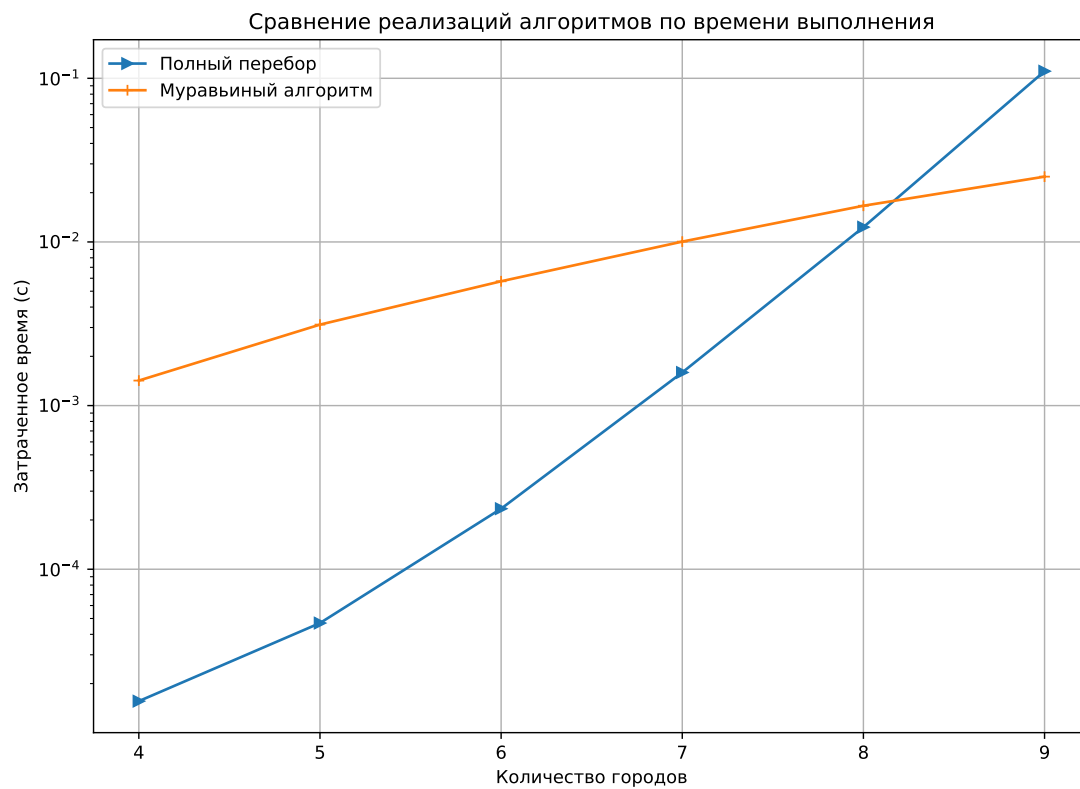


Рисунок 4.2 – Сравнение реализаций алгоритмов по времени выполнения

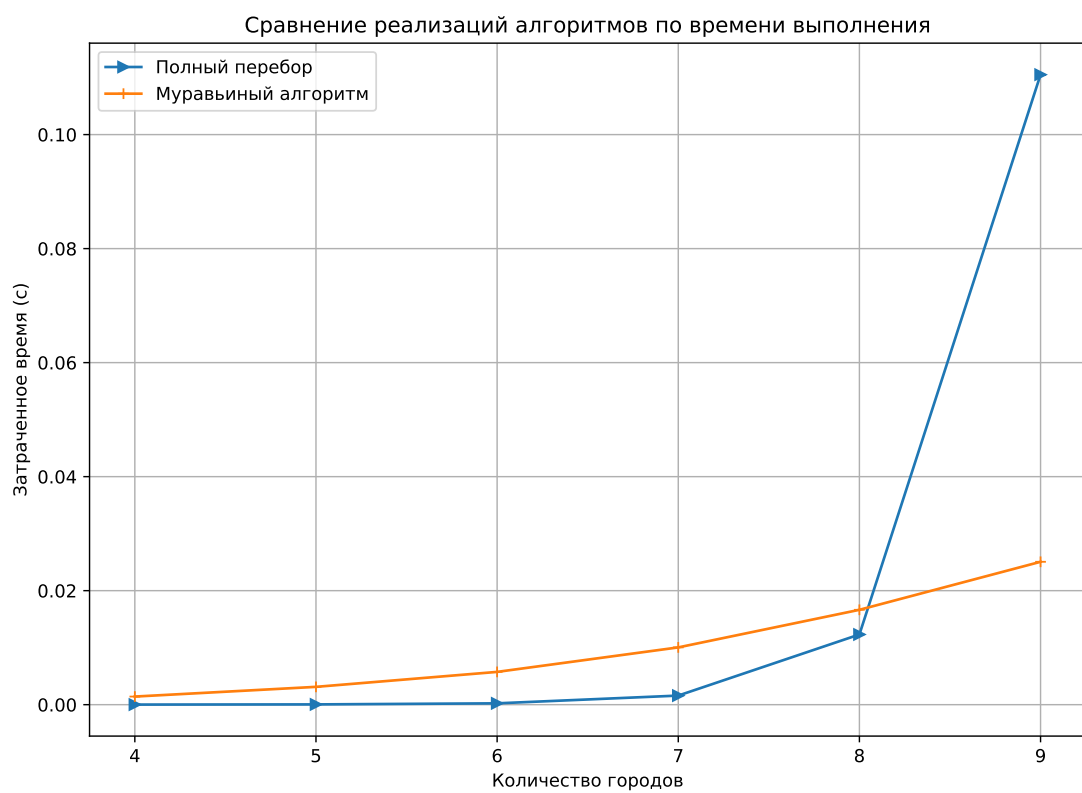


Рисунок 4.3 – Сравнение реализаций алгоритмов по времени выполнения

4.4 Параметризация муравьиного алгоритма

Автоматическая параметризация была проведена для одного класса данных 4.4.1, состоящего из 3 графов. Алгоритм запускался для набора значений $\alpha, \rho \in (0, 1)$ и $t_{max} \in \{5, 25, 50, 100\}$.

Итоговая таблица значений параметризации состоит из следующих колонок:

- α — коэффициент жадности;
- ρ — коэффициент испарения феромона;
- t_{max} — количество дней жизни колонии муравьев;
- *Результат* — максимальная длина пути, полученная муравьиным алгоритмом за 10 запусков;
- *Ошибка* — разность между результатом и эталонным значением.

4.4.1 Класс данных

Класс данных представляет собой набор из 3 орграфов, заданных с помощью матриц смежности (4.1), (4.2), (4.3) размером 10 элементов с равномерным распределением значений весов от 1 до 100.

$$M_1 = \begin{pmatrix} 0 & 34 & 84 & 64 & 37 & 51 & 7 & 55 & 28 & 10 \\ 34 & 0 & 89 & 61 & 26 & 64 & 31 & 82 & 19 & 48 \\ 11 & 16 & 0 & 46 & 2 & 75 & 48 & 65 & 45 & 62 \\ 12 & 30 & 46 & 0 & 71 & 37 & 27 & 70 & 44 & 25 \\ 68 & 20 & 31 & 36 & 0 & 47 & 44 & 72 & 29 & 82 \\ 90 & 78 & 11 & 44 & 91 & 0 & 62 & 43 & 73 & 77 \\ 90 & 33 & 80 & 8 & 98 & 48 & 0 & 99 & 36 & 71 \\ 18 & 16 & 28 & 22 & 99 & 62 & 80 & 0 & 31 & 63 \\ 51 & 77 & 45 & 91 & 45 & 41 & 77 & 40 & 0 & 26 \\ 55 & 67 & 24 & 8 & 57 & 29 & 82 & 50 & 78 & 0 \end{pmatrix} \quad (4.1)$$

$$M_2 = \begin{pmatrix} 0 & 73 & 27 & 98 & 40 & 71 & 11 & 31 & 17 & 14 \\ 5 & 0 & 19 & 99 & 74 & 29 & 43 & 54 & 94 & 58 \\ 78 & 66 & 0 & 17 & 79 & 98 & 74 & 64 & 39 & 45 \\ 4 & 83 & 56 & 0 & 71 & 28 & 32 & 50 & 96 & 11 \\ 36 & 62 & 51 & 35 & 0 & 51 & 62 & 43 & 39 & 94 \\ 6 & 3 & 53 & 23 & 42 & 0 & 10 & 9 & 46 & 10 \\ 25 & 82 & 23 & 50 & 32 & 65 & 0 & 72 & 69 & 65 \\ 53 & 93 & 55 & 62 & 71 & 42 & 79 & 0 & 79 & 78 \\ 65 & 55 & 45 & 17 & 44 & 82 & 68 & 32 & 0 & 48 \\ 93 & 70 & 2 & 53 & 62 & 27 & 76 & 25 & 81 & 0 \end{pmatrix} \quad (4.2)$$

$$M_3 = \begin{pmatrix} 0 & 22 & 97 & 48 & 76 & 64 & 32 & 66 & 63 & 50 \\ 34 & 0 & 69 & 46 & 13 & 63 & 71 & 99 & 48 & 83 \\ 74 & 21 & 0 & 3 & 64 & 49 & 78 & 38 & 61 & 42 \\ 86 & 4 & 40 & 0 & 57 & 16 & 74 & 36 & 99 & 98 \\ 92 & 51 & 98 & 42 & 0 & 38 & 19 & 28 & 19 & 18 \\ 29 & 92 & 47 & 30 & 99 & 0 & 33 & 86 & 51 & 4 \\ 11 & 9 & 94 & 46 & 31 & 5 & 0 & 46 & 55 & 45 \\ 52 & 95 & 13 & 3 & 19 & 25 & 77 & 0 & 75 & 14 \\ 64 & 49 & 25 & 36 & 4 & 96 & 46 & 50 & 0 & 61 \\ 17 & 54 & 68 & 80 & 81 & 84 & 93 & 35 & 94 & 0 \end{pmatrix} \quad (4.3)$$

Результаты параметризации для класса данных содержатся в приложении А.

Вывод

В результате замеров времени выполнения реализаций алгоритмов было выявлено, что реализация алгоритма полного перебора оказалась быстрее реализации муравьиного алгоритма при количестве городов меньше 8, например, при количестве городов 4, реализация алгоритма полного перебора выигрывает реализацию муравьиного алгоритма в 91 раз. Но при количестве городов 9, реализация алгоритма полного перебора оказалась хуже реализации муравьиного алгоритма в 4 раза по времени выполнения. Что соответствует асимптотическим оценкам трудоемкости алгоритмов, а именно алгоритм

полного перебора обладает большей асимптотической оценкой $O(n!)$, чем муравьиный алгоритм $O(Tn^4)$. Т. о. алгоритм полного перебора выгоднее применять при малом количестве городов.

В результате параметризации были подобраны параметры, при которых реализация муравьиного алгоритма показывает наилучшие результаты. Для первого графа такими параметрами являются: $\alpha = 0.1, \rho = \{0.1, 0.3\}, t_{max} = 100$. Для второго: $\alpha = 0.1, \rho = 0.9, t_{max} = 100$. Для третьего:

- $\alpha = 0.1, \rho = 0.1, t_{max} = \{50, 100\}$;
- $\alpha = 0.1, \rho = 0.3, t_{max} = 100$;
- $\alpha = 0.1, \rho = 0.5, t_{max} = \{50, 100\}$;
- $\alpha = 0.1, \rho = 0.7, t_{max} = 100$;
- $\alpha = 0.1, \rho = 0.9, t_{max} = 100$.

Т. о. для реализации муравьиного алгоритма, применяемого к рассмотренному классу данных, рекомендуется использовать параметры: $\alpha = 0.1, \rho = 0.1, t_{max} = 100$.

ЗАКЛЮЧЕНИЕ

Для достижения поставленной цели были выполнены следующие задачи:

- описаны алгоритмы решения задачи коммивояжера;
- спроектировано программное обеспечение, реализующее алгоритмы решения задачи коммивояжера;
- выбраны инструменты для реализации и замера процессорного времени выполнения реализаций решения задачи;
- проанализированы затраты реализаций алгоритмов по времени.

В результате исследования времени выполнения реализаций алгоритмов было выявлено, что реализация алгоритма полного перебора оказалась быстрее реализации муравьиного алгоритма при количестве городов меньше 8, например, при количестве городов 4, реализация алгоритма полного перебора выигрывает реализацию муравьиного алгоритма в 91 раз по времени выполнения. Но при количестве городов 9, реализация алгоритма полного перебора оказалась хуже реализации муравьиного алгоритма в 4 раза по времени выполнения. Что соответствует асимптотическим оценкам трудоемкости алгоритмов, а именно алгоритм полного перебора обладает большей асимптотической оценкой $O(n!)$, чем муравьиный алгоритм $O(Tn^4)$.

В ходе параметризации было выявлено, что лучший результат, реализация муравьиного алгоритма, достигает при значениях параметров: $\alpha = 0.1$, $\rho = 0.1$, $t_{max} = 100$.

Цель данной лабораторной работы была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Володина Е.В. С. Е.* Практическое применение алгоритма решения задачи коммивояжера // ИВД. — 2015. — № 2.
2. *Colormi A., Dorigo M., Maniezzo V.* Distributed Optimization by Ant Colonies // Proceedings of the First European Conference on Artificial Life. — 1991.
3. The official home of the Python Programming Language [Электронный ресурс]. — Режим доступа: <https://www.python.org/> (дата обращения: 19.09.2023).
4. time — Time access and conversions [Электронный ресурс]. — Режим доступа: <https://docs.python.org/3/library/time.html> (дата обращения: 19.09.2023).
5. Amd [Электронный ресурс]. — Режим доступа: <https://www.amd.com/en.html> (дата обращения: 28.09.2023).
6. Windows 10 Pro 22h2 64-bit [Электронный ресурс]. — Режим доступа: <https://www.microsoft.com/ru-ru/software-download/windows10> (дата обращения: 28.09.2023).

ПРИЛОЖЕНИЕ А

Таблица А.1 – Параметризация для матрицы M_1

| α | ρ | t_{max} | Результат | Ошибка |
|----------|--------|-----------|-----------|--------|
| 0.1 | 0.1 | 5 | 234 | 73 |
| 0.1 | 0.1 | 25 | 185 | 24 |
| 0.1 | 0.1 | 50 | 189 | 28 |
| 0.1 | 0.1 | 100 | 176 | 15 |
| 0.1 | 0.3 | 5 | 225 | 64 |
| 0.1 | 0.3 | 25 | 200 | 39 |
| 0.1 | 0.3 | 50 | 184 | 23 |
| 0.1 | 0.3 | 100 | 176 | 15 |
| 0.1 | 0.5 | 5 | 231 | 70 |
| 0.1 | 0.5 | 25 | 200 | 39 |
| 0.1 | 0.5 | 50 | 179 | 18 |
| 0.1 | 0.5 | 100 | 188 | 27 |
| 0.1 | 0.7 | 5 | 254 | 93 |
| 0.1 | 0.7 | 25 | 189 | 28 |
| 0.1 | 0.7 | 50 | 184 | 23 |
| 0.1 | 0.7 | 100 | 180 | 19 |
| 0.1 | 0.9 | 5 | 241 | 80 |
| 0.1 | 0.9 | 25 | 180 | 19 |
| 0.1 | 0.9 | 50 | 185 | 24 |
| 0.1 | 0.9 | 100 | 180 | 19 |
| 0.3 | 0.1 | 5 | 238 | 77 |
| 0.3 | 0.1 | 25 | 214 | 53 |
| 0.3 | 0.1 | 50 | 195 | 34 |
| 0.3 | 0.1 | 100 | 184 | 23 |
| 0.3 | 0.3 | 5 | 234 | 73 |
| 0.3 | 0.3 | 25 | 214 | 53 |
| 0.3 | 0.3 | 50 | 200 | 39 |
| 0.3 | 0.3 | 100 | 186 | 25 |

| | | | | |
|-----|-----|-----|-----|-----|
| 0.3 | 0.5 | 5 | 256 | 95 |
| 0.3 | 0.5 | 25 | 204 | 43 |
| 0.3 | 0.5 | 50 | 204 | 43 |
| 0.3 | 0.5 | 100 | 186 | 25 |
| 0.3 | 0.7 | 5 | 260 | 99 |
| 0.3 | 0.7 | 25 | 204 | 43 |
| 0.3 | 0.7 | 50 | 186 | 25 |
| 0.3 | 0.7 | 100 | 188 | 27 |
| 0.3 | 0.9 | 5 | 270 | 109 |
| 0.3 | 0.9 | 25 | 209 | 48 |
| 0.3 | 0.9 | 50 | 195 | 34 |
| 0.3 | 0.9 | 100 | 193 | 32 |
| 0.5 | 0.1 | 5 | 265 | 104 |
| 0.5 | 0.1 | 25 | 228 | 67 |
| 0.5 | 0.1 | 50 | 222 | 61 |
| 0.5 | 0.1 | 100 | 203 | 42 |
| 0.5 | 0.3 | 5 | 296 | 135 |
| 0.5 | 0.3 | 25 | 242 | 81 |
| 0.5 | 0.3 | 50 | 226 | 65 |
| 0.5 | 0.3 | 100 | 195 | 34 |
| 0.5 | 0.5 | 5 | 266 | 105 |
| 0.5 | 0.5 | 25 | 223 | 62 |
| 0.5 | 0.5 | 50 | 233 | 72 |
| 0.5 | 0.5 | 100 | 209 | 48 |
| 0.5 | 0.7 | 5 | 264 | 103 |
| 0.5 | 0.7 | 25 | 223 | 62 |
| 0.5 | 0.7 | 50 | 204 | 43 |
| 0.5 | 0.7 | 100 | 195 | 34 |
| 0.5 | 0.9 | 5 | 271 | 110 |
| 0.5 | 0.9 | 25 | 225 | 64 |
| 0.5 | 0.9 | 50 | 207 | 46 |
| 0.5 | 0.9 | 100 | 197 | 36 |
| 0.7 | 0.1 | 5 | 283 | 122 |

| | | | | |
|-----|-----|-----|-----|-----|
| 0.7 | 0.1 | 25 | 248 | 87 |
| 0.7 | 0.1 | 50 | 232 | 71 |
| 0.7 | 0.1 | 100 | 218 | 57 |
| 0.7 | 0.3 | 5 | 295 | 134 |
| 0.7 | 0.3 | 25 | 248 | 87 |
| 0.7 | 0.3 | 50 | 225 | 64 |
| 0.7 | 0.3 | 100 | 214 | 53 |
| 0.7 | 0.5 | 5 | 296 | 135 |
| 0.7 | 0.5 | 25 | 243 | 82 |
| 0.7 | 0.5 | 50 | 235 | 74 |
| 0.7 | 0.5 | 100 | 223 | 62 |
| 0.7 | 0.7 | 5 | 288 | 127 |
| 0.7 | 0.7 | 25 | 248 | 87 |
| 0.7 | 0.7 | 50 | 253 | 92 |
| 0.7 | 0.7 | 100 | 215 | 54 |
| 0.7 | 0.9 | 5 | 320 | 159 |
| 0.7 | 0.9 | 25 | 244 | 83 |
| 0.7 | 0.9 | 50 | 244 | 83 |
| 0.7 | 0.9 | 100 | 224 | 63 |
| 0.9 | 0.1 | 5 | 334 | 173 |
| 0.9 | 0.1 | 25 | 293 | 132 |
| 0.9 | 0.1 | 50 | 254 | 93 |
| 0.9 | 0.1 | 100 | 242 | 81 |
| 0.9 | 0.3 | 5 | 334 | 173 |
| 0.9 | 0.3 | 25 | 282 | 121 |
| 0.9 | 0.3 | 50 | 256 | 95 |
| 0.9 | 0.3 | 100 | 256 | 95 |
| 0.9 | 0.5 | 5 | 342 | 181 |
| 0.9 | 0.5 | 25 | 267 | 106 |
| 0.9 | 0.5 | 50 | 259 | 98 |
| 0.9 | 0.5 | 100 | 237 | 76 |
| 0.9 | 0.7 | 5 | 327 | 166 |
| 0.9 | 0.7 | 25 | 279 | 118 |

| | | | | |
|-----|-----|-----|-----|-----|
| 0.9 | 0.7 | 50 | 259 | 98 |
| 0.9 | 0.7 | 100 | 240 | 79 |
| 0.9 | 0.9 | 5 | 320 | 159 |
| 0.9 | 0.9 | 25 | 276 | 115 |
| 0.9 | 0.9 | 50 | 254 | 93 |
| 0.9 | 0.9 | 100 | 246 | 85 |

Таблица А.2 – Параметризация для матрицы M_2

| α | ρ | t_{max} | Результат | Ошибка |
|----------|--------|-----------|-----------|--------|
| 0.1 | 0.1 | 5 | 264 | 65 |
| 0.1 | 0.1 | 25 | 248 | 49 |
| 0.1 | 0.1 | 50 | 223 | 24 |
| 0.1 | 0.1 | 100 | 236 | 37 |
| 0.1 | 0.3 | 5 | 280 | 81 |
| 0.1 | 0.3 | 25 | 256 | 57 |
| 0.1 | 0.3 | 50 | 231 | 32 |
| 0.1 | 0.3 | 100 | 221 | 22 |
| 0.1 | 0.5 | 5 | 250 | 51 |
| 0.1 | 0.5 | 25 | 233 | 34 |
| 0.1 | 0.5 | 50 | 236 | 37 |
| 0.1 | 0.5 | 100 | 221 | 22 |
| 0.1 | 0.7 | 5 | 303 | 104 |
| 0.1 | 0.7 | 25 | 237 | 38 |
| 0.1 | 0.7 | 50 | 227 | 28 |
| 0.1 | 0.7 | 100 | 234 | 35 |
| 0.1 | 0.9 | 5 | 278 | 79 |
| 0.1 | 0.9 | 25 | 248 | 49 |
| 0.1 | 0.9 | 50 | 226 | 27 |
| 0.1 | 0.9 | 100 | 220 | 21 |
| 0.3 | 0.1 | 5 | 282 | 83 |
| 0.3 | 0.1 | 25 | 261 | 62 |

| | | | | |
|-----|-----|-----|-----|-----|
| 0.3 | 0.1 | 50 | 236 | 37 |
| 0.3 | 0.1 | 100 | 231 | 32 |
| 0.3 | 0.3 | 5 | 294 | 95 |
| 0.3 | 0.3 | 25 | 258 | 59 |
| 0.3 | 0.3 | 50 | 248 | 49 |
| 0.3 | 0.3 | 100 | 240 | 41 |
| 0.3 | 0.5 | 5 | 285 | 86 |
| 0.3 | 0.5 | 25 | 267 | 68 |
| 0.3 | 0.5 | 50 | 242 | 43 |
| 0.3 | 0.5 | 100 | 237 | 38 |
| 0.3 | 0.7 | 5 | 286 | 87 |
| 0.3 | 0.7 | 25 | 261 | 62 |
| 0.3 | 0.7 | 50 | 252 | 53 |
| 0.3 | 0.7 | 100 | 225 | 26 |
| 0.3 | 0.9 | 5 | 292 | 93 |
| 0.3 | 0.9 | 25 | 258 | 59 |
| 0.3 | 0.9 | 50 | 251 | 52 |
| 0.3 | 0.9 | 100 | 228 | 29 |
| 0.5 | 0.1 | 5 | 318 | 119 |
| 0.5 | 0.1 | 25 | 268 | 69 |
| 0.5 | 0.1 | 50 | 259 | 60 |
| 0.5 | 0.1 | 100 | 240 | 41 |
| 0.5 | 0.3 | 5 | 317 | 118 |
| 0.5 | 0.3 | 25 | 260 | 61 |
| 0.5 | 0.3 | 50 | 262 | 63 |
| 0.5 | 0.3 | 100 | 239 | 40 |
| 0.5 | 0.5 | 5 | 325 | 126 |
| 0.5 | 0.5 | 25 | 272 | 73 |
| 0.5 | 0.5 | 50 | 263 | 64 |
| 0.5 | 0.5 | 100 | 242 | 43 |
| 0.5 | 0.7 | 5 | 306 | 107 |
| 0.5 | 0.7 | 25 | 277 | 78 |
| 0.5 | 0.7 | 50 | 240 | 41 |

| | | | | |
|-----|-----|-----|-----|-----|
| 0.5 | 0.7 | 100 | 231 | 32 |
| 0.5 | 0.9 | 5 | 300 | 101 |
| 0.5 | 0.9 | 25 | 259 | 60 |
| 0.5 | 0.9 | 50 | 264 | 65 |
| 0.5 | 0.9 | 100 | 251 | 52 |
| 0.7 | 0.1 | 5 | 336 | 137 |
| 0.7 | 0.1 | 25 | 284 | 85 |
| 0.7 | 0.1 | 50 | 273 | 74 |
| 0.7 | 0.1 | 100 | 242 | 43 |
| 0.7 | 0.3 | 5 | 309 | 110 |
| 0.7 | 0.3 | 25 | 297 | 98 |
| 0.7 | 0.3 | 50 | 270 | 71 |
| 0.7 | 0.3 | 100 | 255 | 56 |
| 0.7 | 0.5 | 5 | 328 | 129 |
| 0.7 | 0.5 | 25 | 304 | 105 |
| 0.7 | 0.5 | 50 | 269 | 70 |
| 0.7 | 0.5 | 100 | 257 | 58 |
| 0.7 | 0.7 | 5 | 330 | 131 |
| 0.7 | 0.7 | 25 | 295 | 96 |
| 0.7 | 0.7 | 50 | 263 | 64 |
| 0.7 | 0.7 | 100 | 277 | 78 |
| 0.7 | 0.9 | 5 | 349 | 150 |
| 0.7 | 0.9 | 25 | 280 | 81 |
| 0.7 | 0.9 | 50 | 279 | 80 |
| 0.7 | 0.9 | 100 | 249 | 50 |
| 0.9 | 0.1 | 5 | 373 | 174 |
| 0.9 | 0.1 | 25 | 305 | 106 |
| 0.9 | 0.1 | 50 | 282 | 83 |
| 0.9 | 0.1 | 100 | 270 | 71 |
| 0.9 | 0.3 | 5 | 369 | 170 |
| 0.9 | 0.3 | 25 | 319 | 120 |
| 0.9 | 0.3 | 50 | 299 | 100 |
| 0.9 | 0.3 | 100 | 279 | 80 |

| | | | | |
|-----|-----|-----|-----|-----|
| 0.9 | 0.5 | 5 | 346 | 147 |
| 0.9 | 0.5 | 25 | 307 | 108 |
| 0.9 | 0.5 | 50 | 284 | 85 |
| 0.9 | 0.5 | 100 | 286 | 87 |
| 0.9 | 0.7 | 5 | 363 | 164 |
| 0.9 | 0.7 | 25 | 308 | 109 |
| 0.9 | 0.7 | 50 | 284 | 85 |
| 0.9 | 0.7 | 100 | 284 | 85 |
| 0.9 | 0.9 | 5 | 331 | 132 |
| 0.9 | 0.9 | 25 | 307 | 108 |
| 0.9 | 0.9 | 50 | 297 | 98 |
| 0.9 | 0.9 | 100 | 273 | 74 |

Таблица А.3 – Параметризация для матрицы M_3

| α | ρ | t_{max} | Результат | Ошибка |
|----------|--------|-----------|-----------|--------|
| 0.1 | 0.1 | 5 | 223 | 95 |
| 0.1 | 0.1 | 25 | 186 | 58 |
| 0.1 | 0.1 | 50 | 140 | 12 |
| 0.1 | 0.1 | 100 | 140 | 12 |
| 0.1 | 0.3 | 5 | 233 | 105 |
| 0.1 | 0.3 | 25 | 175 | 47 |
| 0.1 | 0.3 | 50 | 156 | 28 |
| 0.1 | 0.3 | 100 | 140 | 12 |
| 0.1 | 0.5 | 5 | 237 | 109 |
| 0.1 | 0.5 | 25 | 171 | 43 |
| 0.1 | 0.5 | 50 | 140 | 12 |
| 0.1 | 0.5 | 100 | 140 | 12 |
| 0.1 | 0.7 | 5 | 233 | 105 |
| 0.1 | 0.7 | 25 | 171 | 43 |
| 0.1 | 0.7 | 50 | 180 | 52 |

| | | | | |
|-----|-----|-----|-----|-----|
| 0.1 | 0.7 | 100 | 140 | 12 |
| 0.1 | 0.9 | 5 | 220 | 92 |
| 0.1 | 0.9 | 25 | 192 | 64 |
| 0.1 | 0.9 | 50 | 159 | 31 |
| 0.1 | 0.9 | 100 | 140 | 12 |
| 0.3 | 0.1 | 5 | 233 | 105 |
| 0.3 | 0.1 | 25 | 202 | 74 |
| 0.3 | 0.1 | 50 | 186 | 58 |
| 0.3 | 0.1 | 100 | 166 | 38 |
| 0.3 | 0.3 | 5 | 247 | 119 |
| 0.3 | 0.3 | 25 | 196 | 68 |
| 0.3 | 0.3 | 50 | 171 | 43 |
| 0.3 | 0.3 | 100 | 173 | 45 |
| 0.3 | 0.5 | 5 | 255 | 127 |
| 0.3 | 0.5 | 25 | 210 | 82 |
| 0.3 | 0.5 | 50 | 172 | 44 |
| 0.3 | 0.5 | 100 | 156 | 28 |
| 0.3 | 0.7 | 5 | 240 | 112 |
| 0.3 | 0.7 | 25 | 193 | 65 |
| 0.3 | 0.7 | 50 | 186 | 58 |
| 0.3 | 0.7 | 100 | 174 | 46 |
| 0.3 | 0.9 | 5 | 283 | 155 |
| 0.3 | 0.9 | 25 | 222 | 94 |
| 0.3 | 0.9 | 50 | 193 | 65 |
| 0.3 | 0.9 | 100 | 150 | 22 |
| 0.5 | 0.1 | 5 | 289 | 161 |
| 0.5 | 0.1 | 25 | 228 | 100 |
| 0.5 | 0.1 | 50 | 213 | 85 |
| 0.5 | 0.1 | 100 | 205 | 77 |
| 0.5 | 0.3 | 5 | 263 | 135 |
| 0.5 | 0.3 | 25 | 225 | 97 |
| 0.5 | 0.3 | 50 | 209 | 81 |
| 0.5 | 0.3 | 100 | 180 | 52 |

| | | | | |
|-----|-----|-----|-----|-----|
| 0.5 | 0.5 | 5 | 285 | 157 |
| 0.5 | 0.5 | 25 | 237 | 109 |
| 0.5 | 0.5 | 50 | 206 | 78 |
| 0.5 | 0.5 | 100 | 197 | 69 |
| 0.5 | 0.7 | 5 | 288 | 160 |
| 0.5 | 0.7 | 25 | 218 | 90 |
| 0.5 | 0.7 | 50 | 213 | 85 |
| 0.5 | 0.7 | 100 | 190 | 62 |
| 0.5 | 0.9 | 5 | 271 | 143 |
| 0.5 | 0.9 | 25 | 207 | 79 |
| 0.5 | 0.9 | 50 | 208 | 80 |
| 0.5 | 0.9 | 100 | 186 | 58 |
| 0.7 | 0.1 | 5 | 297 | 169 |
| 0.7 | 0.1 | 25 | 232 | 104 |
| 0.7 | 0.1 | 50 | 212 | 84 |
| 0.7 | 0.1 | 100 | 209 | 81 |
| 0.7 | 0.3 | 5 | 273 | 145 |
| 0.7 | 0.3 | 25 | 243 | 115 |
| 0.7 | 0.3 | 50 | 238 | 110 |
| 0.7 | 0.3 | 100 | 217 | 89 |
| 0.7 | 0.5 | 5 | 311 | 183 |
| 0.7 | 0.5 | 25 | 273 | 145 |
| 0.7 | 0.5 | 50 | 230 | 102 |
| 0.7 | 0.5 | 100 | 215 | 87 |
| 0.7 | 0.7 | 5 | 285 | 157 |
| 0.7 | 0.7 | 25 | 244 | 116 |
| 0.7 | 0.7 | 50 | 211 | 83 |
| 0.7 | 0.7 | 100 | 214 | 86 |
| 0.7 | 0.9 | 5 | 291 | 163 |
| 0.7 | 0.9 | 25 | 231 | 103 |
| 0.7 | 0.9 | 50 | 233 | 105 |
| 0.7 | 0.9 | 100 | 194 | 66 |
| 0.9 | 0.1 | 5 | 320 | 192 |

| | | | | |
|-----|-----|-----|-----|-----|
| 0.9 | 0.1 | 25 | 280 | 152 |
| 0.9 | 0.1 | 50 | 259 | 131 |
| 0.9 | 0.1 | 100 | 240 | 112 |
| 0.9 | 0.3 | 5 | 321 | 193 |
| 0.9 | 0.3 | 25 | 272 | 144 |
| 0.9 | 0.3 | 50 | 253 | 125 |
| 0.9 | 0.3 | 100 | 242 | 114 |
| 0.9 | 0.5 | 5 | 325 | 197 |
| 0.9 | 0.5 | 25 | 278 | 150 |
| 0.9 | 0.5 | 50 | 237 | 109 |
| 0.9 | 0.5 | 100 | 227 | 99 |
| 0.9 | 0.7 | 5 | 322 | 194 |
| 0.9 | 0.7 | 25 | 249 | 121 |
| 0.9 | 0.7 | 50 | 242 | 114 |
| 0.9 | 0.7 | 100 | 234 | 106 |
| 0.9 | 0.9 | 5 | 320 | 192 |
| 0.9 | 0.9 | 25 | 257 | 129 |
| 0.9 | 0.9 | 50 | 242 | 114 |
| 0.9 | 0.9 | 100 | 236 | 108 |