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»

Отчёт

по лабораторной работе №6

Название: Муравьиный алгоритм

Дисциплина: Анализ алгоритмов

Студент ИУ7-54Б _____ Л.Е.Тартыков

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Л.Л. Волкова

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Содержание

Введение

Пути возможного решения NP-полных задач интересуют математиков со всего мира уже не одно десятилетие. Они пытаются понять, можно ли решить такие задачи за полиномиальное время. Одним из таких решений является использование эвристических алгоритмов, которые не гарантируют правильность решения задачи, но при верном подборе параметров позволяют обеспечить хорошее качество решения.

В последние два десятилетия при оптимизации сложных систем для поиска наилучших решений исследователи стали применять природные механизмы. Одним из предложенных решений таких задач является практическое применение муравьиных алгоритмов, которые основываются на поведении колонии муравьев[1]. Такой природных механизм позволяет решить такие задачи, как “задача коммивояжера“, а также решения аналогичных задач поиска маршрутов на графах.

Целью лабораторной работы является изучение предложенной оптимизации и подбор параметров наилучшего решения для двух классов данных. Для её достижения необходимо выполнить следующие задачи:

- исследовать подходы алгоритмов полного перебора и оптимизации подражанием муравьиной колонии для решения задачи коммивояжера;
- привести схемы этих алгоритмов;
- описать классы эквивалентности тестов, необходимые для параметризации алгоритма муравьев;
- выполнить параметризацию муравьиного алгоритма на основе разработанных классов эквивалентности;
- провести сравнительный анализ этих алгоритмов на качество решения задачи и времени выполнения.

1 Аналитический раздел

В данном разделе описывается задача коммивояжера. Сформулированы пути решения методом "грубой силы"и муравьиной колонии. Описана их математическая модель.

1.1 Задача коммивояжёра

Коммивояжёр (фр. commis voyageur) — бродячий торговец.

Является важной задачей, относящейся к транспортной логистике. Она формулируется следующим образом: "Рассматриваются N городов и попарное расстояние между каждым из них. Требуется найти такой порядок посещения городов, чтобы суммарное пройденное расстояние было минимальным, а каждый город посещался ровно один раз, при этом коммивояжер должен вернуться в тот город, откуда начал свой маршрут".

Математическую модель данной задачи можно описать следующим образом:

- города и связь между ними можно представить в виде графа. Таким образом, вершины графа соответствуют городам, а ребра между вершинами - путям между сообщениями.

В общем случае для гарантии существования маршрута принято считать, что граф является полностью связным - между произвольной парой вершин существует ребро.

1.2 Алгоритм полного перебора

Данную задачу гарантированно можно решить полным перебором (методом "грубой силы") всех возможных вариантов и последующим выбором оптимального. Для каждого такого варианта строится свой маршрут и определяется его длина. Происходит сравнение значения с локальным минимумом и вычисляется кратчайший.

При таком подходе это число различных, неповторяющихся комбинаций зависит от общего числа городов n и составляет $n!$, а значит время, затрачиваемое на полный перебор, растёт экспоненциально. При числе вершин графа, большем 10, вычисления могут занят часы, годы или даже столетия.

1.3 Алгоритм муравьев

1.3.1 Принципы поведения муравьев

Подход к оптимизации решения путем полного перебора основывается на принципе поведения муравьев. В этой основе лежит самоорганизация, которая позволяет обеспечить достижение общей цели колонии на уровне низкоуровневого взаимодействия. Важной особенностью такого взаимодействия является использование только локальной информации; исключается централизованное управление. В основе самоорганизации лежат результаты взаимодействия следующих компонентов:

- случайность;
- многократность;
- положительная обратная связь;
- отрицательная обратная связь - испарение феромона.

Для передачи информации муравьи могут использовать непрямой способ обмена - стигмержи. Это разнесенный во времени тип взаимодействия, когда один субъект взаимодействия изменяет некоторую часть окружающей среды, а остальные используют информацию об ее состоянии позже, когда находятся в ее окрестности. Биологически стигмержи осуществляется через феромон - является достаточно стойким веществом; чем выше концентрация его концентрация на тропе, тем больше муравьев будет двигаться по ней. Со временем феромон испаряется, что позволяет адаптировать свое поведение под изменения внешней среды.

1.3.2 Муравьиный подход к решению задачи

Моделирование поведения муравьев связано с распределением феромона на тропе — ребре графа в задаче коммивояжера[2]. При этом вероятность включения ребра в маршрут отдельного муравья пропорциональна количеству феромона на этом ребре, а количество откладываемого феромона пропорционально длине маршрута. Чем короче маршрут, тем больше феромона будет отложено на его ребрах, следовательно, большее количество муравьев будет включать его в синтез собственных маршрутов. Моделирование такого подхода, использующего только положительную обратную связь, приводит к преждевременной сходимости — большинство муравьев двигается по локально оптимальному маршруту. Избежать, этого можно, моделируя отрицательную обратную связь в виде испарения феромона. При этом если феромон испаряется быстро, то это приводит к потере памяти колонии и забыванию хороших решений, с другой стороны, большое время испарения может привести к получению устойчивого локально оптимального решения. Теперь, с учетом особенностей задачи коммивояжера, мы можем описать локальные правила поведения муравьев при выборе пути.

- муравьи имеют собственную «память». Поскольку каждый город может быть посещеи только один раз, у каждого муравья есть список уже посещенных городов — список запретов. Обозначим через $J_{i,k}$ список городов, которые необходимо посетить муравью k, находящемуся в городе i;

- муравьи обладают «зрением» — видимость есть эвристическое желание посетить город j, если муравей находится в городе i. Будем считать, что видимость обратно пропорциональна расстоянию между городами i и j — D_{ij} по формуле (1.1):

$$\eta_{ij} = \frac{1}{D_{ij}} \tag{1.1}$$

- муравьи обладают «обонянием» — они могут улавливать след феромона, подтверждающий желание посетить город j из города i, на

основании опыта других муравьев. Количество феромона на ребре (i, j) в момент времени t обозначим через $\tau_{ij}(t)$.

На этом формулируется вероятностно-пропорциональное правило, определяющее вероятность перехода k-ого муравья из города i в город j по формуле (1.2):

$$P_{i,j,k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in J_{i,k}} [\tau_{il}(t)]^{\alpha} * [\eta_{il}]^{\beta}}, & j \in J_{i,k}; \\ 0, & j \notin J_{i,k}, \end{cases} \tag{1.2}$$

где α, β — параметры, задающие веса следа феромона, при $\alpha = 0$ алгоритм вырождается до жадного алгоритма (будет выбран ближайший город). Выбор города является вероятностным, правило 1.2 лишь определяет ширину зоны города j; в общую зону всех городов $J_{i,k}$;, бросается случайное число, которое и определяет выбор муравья. Правило по формуле 1.2 не изменяется в ходе алгоритма, но у двух разных муравьев значение вероятности перехода будут отличаться, т. к. они имеют разный список разрешенных городов.

Пройдя ребро (i, j), муравей откладывает на нем некоторое количество феромона, которое должно быть связано с оптимальностью сделанного выбора. Пусть $\tau_k(t)$ есть маршрут, пройденный муравьем k к моменту времени t, а $L_k(t)$ — длина этого маршрута. Пусть также Q — параметр, имеющий значение порядка длины оптимального пути. Тогда откладываемое количество феромона может быть задано по формуле (1.3):

$$\Delta\tau_{i,j,k}(t) = \begin{cases} \frac{Q}{L_k(t)}, & (i, j) \in T_k(t); \\ 0, & (i, j) \notin T_k(t). \end{cases} \tag{1.3}$$

Правила внешней среды определяют, в первую очередь, испарение феромона. Пусть $\rho \in [0, 1]$ есть коэффициент испарения, тогда правило испарения имеет вид по формуле (1.4):

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}(t); \Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij,k}(t); \tag{1.4}$$

где m — количество муравьев в колонии.

В начале алгоритма количество феромона на ребрах принимается равным небольшому положительному числу. Общее количество муравьев остается постоянным и равным количеству городов, каждый муравей начинает маршрут из своего города. Подбор таких параметров, при которых алгоритм выдает верное решение называется параметризацией.

1.4 Вывод

В данном разделе была описана задача коммивояжера. Сформулированы пути решения методом "грубой силы"и муравьиной колонии. Описана их математическая модель.

Разрабатываемое программное обеспечение должно реализовывать эти два подхода к решению задачи. На вход программе подается симметричная матрица смежностей, а также регулируемые параметры. Программа должна выдавать результат - длину кратчайшего пути и способ обхода всех городов.

2 Конструкторский раздел

В данном разделе представлены описание работы алгоритма; описание структур данных, используемых в алгоритме; описание способа тестирования и выделенных классов эквивалентности; описание памяти, используемой алгоритмом; структура программного обеспечения.

2.1 Описание работы алгоритмов

На рисунке 2.1 представлена схема алгоритма полного перебора.

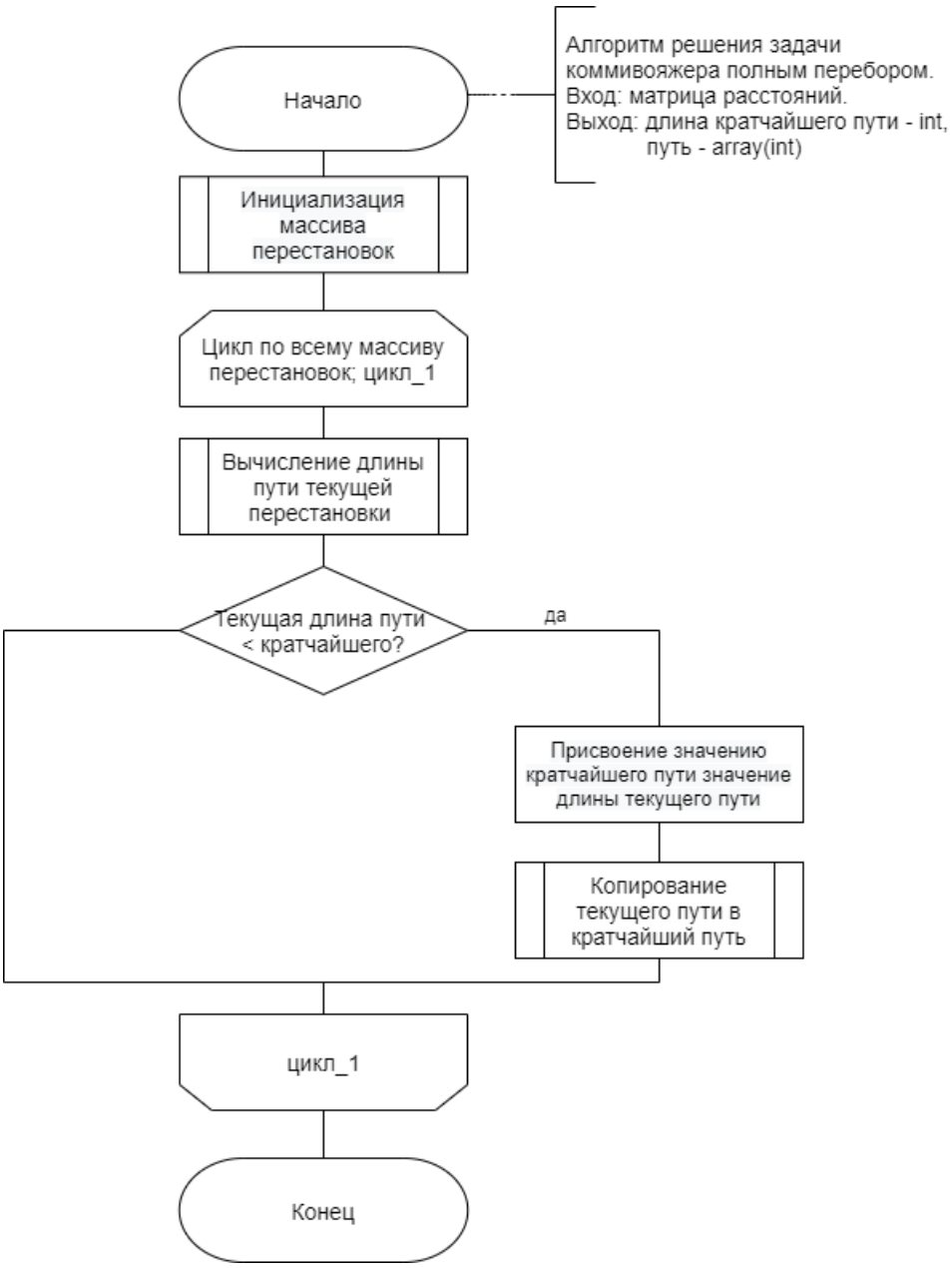

Рисунок 2.1 – Схема алгоритма полного перебора.

На рисунке 2.2 и 2.3 представлена схема алгоритма муравьев.

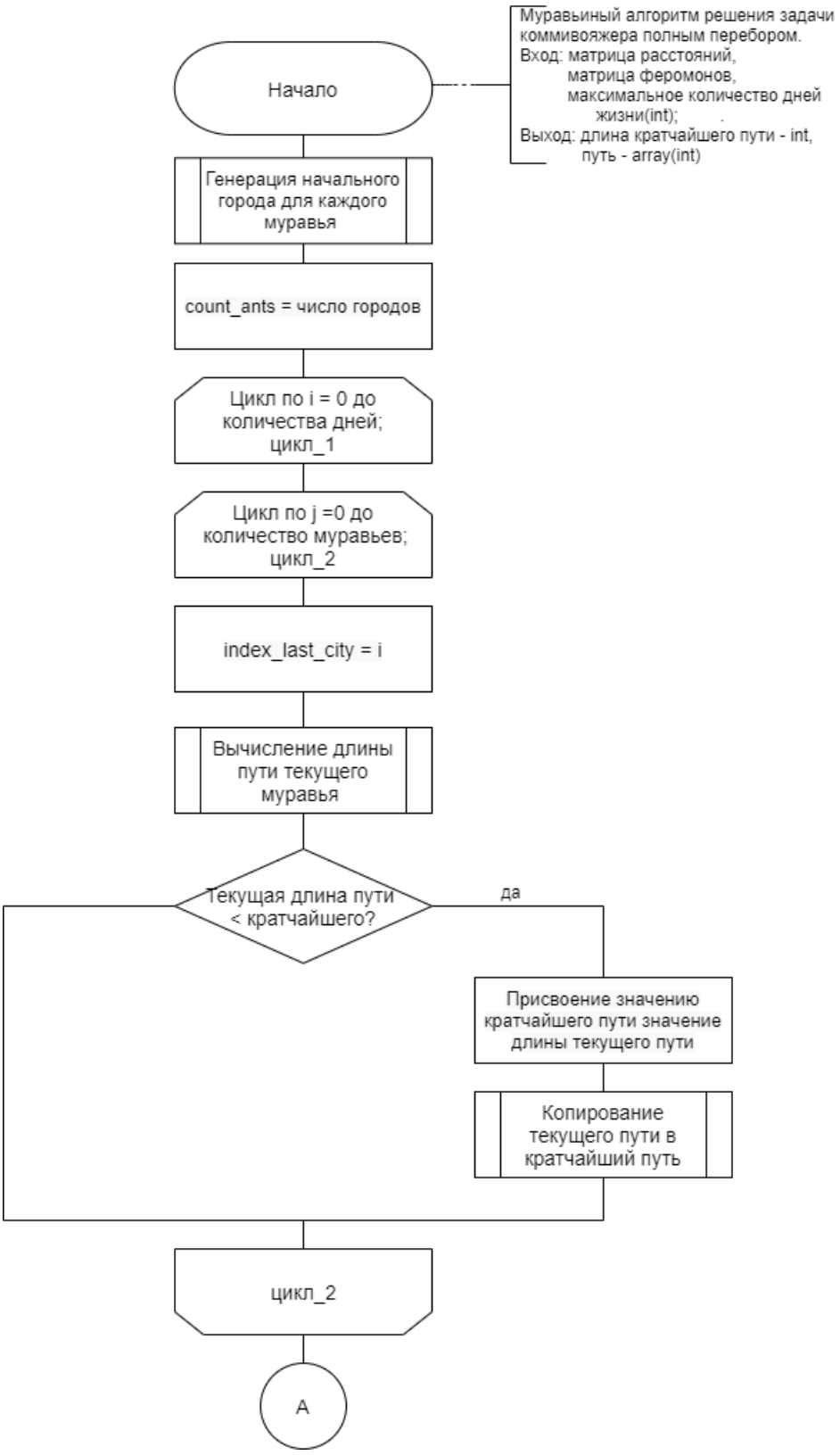

Рисунок 2.2 – Схема алгоритма муравьев.

Рисунок 2.3 – Схема алгоритма муравьев.

2.2 Описание структур данных

Муравьиная колония представлена структурой данных, которая содержит в себе следующие параметры:

- количество дней жизни колонии;
- матрица феромонов, которая содержит информацию о количестве феромона в каждом узле;
- матрица привлекательности - содержит значение привлекательности маршрута для каждого узла;
- мета информация, которая содержит параметры;
- массив муравьев.

Массив муравьев представляет собой количество муравьев, равное количеству городов, содержит следующие параметры

- посещенные города - те города, которые муравей уже посетил;

- количество посещенных городов;
- длина пройденного пути.

Мета информация представляет собой структуру, которая содержит следующие параметры:

- коэффициент испарения феромонов;
- α - коэффициент жадности;
- β - коэффициент привлекательности;
- Q - коэффициент, который используется для нормирования;
- начальное значение феромонов, отличное от нуля.

2.3 Описание способа тестирования и выделенных классов эквивалентности

Тестирование программного обеспечения будет выполнено, используя матрицу смежности, а также набор регулируемых параметров. Матрица является симметричной, диагональные элементы равны нулю.

В качестве классов эквивалентности можно выделить стоимость пути между городами: порядка 50-100 и 500-1000 условных единиц.

2.4 Описание памяти, используемой алгоритмом

2.4.1 Алгоритм полного перебора

Память, необходимая для реализации метода полного перебора, состоит из нескольких параметров, представленных в формуле (2.1):

$$M_{brute} = M_{graph} + M_{cities} \tag{2.1}$$

где:

- M_{graph} - матрица стоимостей - формула (2.2):

$$M_{graph} = size(int) \cdot size(graph\_row) \cdot size(graph\_column); \quad (2.2)$$

- M_{cities} - массив посещенных городов - формула (2.3):

$$M_{cities} = size(int) \cdot (count\_cities - 1); \quad (2.3)$$

2.4.2 Муравьиный алгоритм

Память, используемая муравьиным алгоритмом, складывается как совокупность следующих параметров, представленных в формуле (2.4):

$$M_{colony\_ant} = M_{ants} + M_{max\_days} + M_{meta} + M_{pheromon} + M_{attractiveness} \quad (2.4)$$

где:

- M_{ants} - массив муравьев - формула (2.5):

$$M_{ants} = count\_ants \cdot (size(int) \cdot count\_cities + size(int) \cdot size(double)) \quad (2.5)$$

- M_{max_days} - время жизни колонии - формула (2.6):

$$M_{max\_days} = size(int) \quad (2.6)$$

- M_{meta} - мета информация - формула (2.7):

$$M_{meta} = size(double) \cdot 5 \quad (2.7)$$

- $M_{attractiveness}$ - матрица феромонов - формула (2.8):

$$M_{pheromon} = size(double) \cdot size(pheromon\_row) \cdot size(pheromon\_column) \quad (2.8)$$

- $M_{attractiveness\_memory\_ant}$ - матрица привлекательности - формула (2.9):

$$M_{attractiveness} = size(int) \cdot (count\_cities - 1) \tag{2.9}$$

2.5 Структура программного обеспечения

В качестве парадигмы было использованы структурное программирование. Программное обеспечение состоит из нескольких модулей:

- main - главный модуль, который выполняет вызов функций решения задачи коммивояжера (полный перебор и муравьиный алгоритм);
- brute_force - модуль, содержащий необходимые структуры данных и реализацию метода полного перебора;
- ant - модуль, содержащий необходимые структуры данных и реализацию муравьиного алгоритма;
- cities - модуль, описывающий необходимые структуры и функции для создания массива городов;
- file - модуль, необходимый для считывания данных из файла;
- matrix - модуль, содержащий необходимые структуры и функции для создания матриц.

2.6 Вывод

Были описаны алгоритмы полного перебора и муравьиным методом. Были описаны структуры данных, необходимые для реализации программного обеспечения. Были выделены классы эквивалентности и описан способ тестирования.

3 Технологический раздел

В данном разделе приведены листинги реализованных алгоритмов, требование к программному обеспечению и его тестирование.

3.1 Требования к программному обеспечению

Программа должна удовлетворять следующим требованиям:

- на вход программе подается симметричная матрица стоимостей графа, количество дней жизни колонии, а также регулируемые параметры;
- на вход программе подаются корректные данные;
- на вход программа выдает массив кратчайшего пути и его длину этого пути;

3.2 Выбор средств реализации

Для реализации алгоритмов в данной лабораторной работе был выбран язык программирования C++ [3]. Данный выбор обусловен тем, что он является компилируемым, что увеличивает скорость выполнения программ по сравнению с интерпретируемыми. В качестве среды разработки был использован Visual Studio Code[4], так как в нем присутствуют инструменты для удобства написания и отладки кода. Выполнение замера процессорного времени осуществлено при помощи библиотеки chrono [5].

3.3 Листинги программ

3.3.1 Полный перебор

Ниже представлены листинги разработанного модуля метода полного перебора.

Листинг 3.1 – Программный код алгоритма полного перебора.

```
int find_short_way_by_brute_force(array_t *cities, int **matrix)
{
    output_cities(cities);
    int short_length = 0, min_short_length = 0;
    int i = 0, min_i = 0;

    short_length = find_way(cities, matrix);
    min_short_length = short_length;
    i++;

    while (find_next_permutation(cities) == true)
    {
        output_cities(cities);
        short_length = find_way(cities, matrix);
        if (short_length < min_short_length){
            min_short_length = short_length;
            min_i = i;
        }

        i++;
    }

    return min_short_length;
}
```

3.3.2 Муравьиный алгоритм

На листинге представлен разработанного модуль муравьиного алгоритма:

Листинг 3.2 – Программный код муравьиного алгоритма.

```
void find_short_way_by_ant_algorithm(short_route_t &shortest_route,
    matrix_int_t *path,
ant_colony_t *colony)
{
    int ant_route_length = 0;

    (*colony).meta_data_colony.Q = calculate_Q(path);

    int t_max = (*colony).t_max;
    int count_ants = (*path).size_row;

    generate_start_city_for_ants(colony->ants, count_ants);
    int short_ant_route = (*colony).meta_data_colony.Q*2;
    int index_last_city = 0;

    for (int t = 0; t < t_max; t++)
    {
        for (int i = 0; i < count_ants; i++)
        {
            index_last_city = i;
            ant_route_length = get_route_from_ant(&colony->ants[i], &
                colony->pheromon,
            &colony->attractiveness, &colony->meta_data_colony, path,
            index_last_city);

            if (ant_route_length < short_ant_route)
            {
                shortest_route.length_short_route = ant_route_length;
                copy_ant_route_to_short(shortest_route, colony->ants[i].
                    visit_cities);
            }
        }

        add_pheromone_route(colony, (*path).size_row);
        evaporate_pheromon(&colony->pheromon, colony->meta_data_colony.
            koeff_evaporation,
        colony->meta_data_colony.start_evaporation);
        reset_ant_to_default(&colony->ants[i]);
    }
}
```

3.4 Вспомогательные модули

3.4.1 Полный перебор

На листингах представлены вспомогательные программные модули, которые используются в алгоритме полного перебора:

Листинг 3.3 – Программный код нахождения следующей перестановки.

```
bool find_next_permutation(array_t *cities)
{
    int j = (*cities).size_array - 2;
    while (j != -1 && (*cities).array[j] >= (*cities).array[j + 1]){
        j--;
    }

    if (j == -1)
    return false;

    int k = (*cities).size_array - 1;
    while ((*cities).array[j] >= (*cities).array[k]){
        k--;
    }

    swap_elems_array(cities->array, j, k);
    int left = j + 1, right = (*cities).size_array - 1;

    while (left < right){
        swap_elems_array(cities->array, left++, right--);
    }

    return true;
}
```

3.4.2 Муравьиный алгоритм

На листингах представлены вспомогательные программные модули, которые используются в муравьином алгоритме:

Листинг 3.4 – Программный код вычисления параметра Q.

```
int calculate_Q(matrix_int_t *path)
{
    int size_row = (*path).size_row;
    int size_column = (*path).size_column;
    int average_length = 0;

    for (int i = 0; i < size_row; i++){
        for (int j = 0; j < i; j++){
            average_length += (*path).matrix[i][j];
        }
    }

    average_length /= 2;
    return average_length;
}
```

Листинг 3.5 – Программный код вычисления маршрута каждого муравья.

```
int get_route_from_ant(ant_t *ant, matrix_double_t *pheromons,
matrix_double_t *attractivness, meta_t *meta_data_colony,
matrix_int_t *paths, int index_last_city)
{
    int chosen_city;
    int count_cities = ant->visit_cities->size_array;
    double get_way_length = 0;

    (*ant).count_visited_cities += 1;
    int last_index_last_city = index_last_city;

    for (int i = 0; i < count_cities - 1; i++)
    {
        chosen_city = choose_next_city(&get_way_length, ant, paths,
           pheromons,
        attractivness, last_index_last_city, *meta_data_colony);
        last_index_last_city = chosen_city;
        (*ant).visit_cities->array[i+1] = chosen_city;
        (*ant).count_visited_cities += 1;
        (*ant).visited_way_length += get_way_length;
    }

    int last_visited_city = (*ant).visit_cities->array[count_cities -
       1];

    (*ant).visited_way_length += paths->matrix[last_index_last_city][
       index_last_city];
    return (*ant).visited_way_length;
```

```
26 }
```

Листинг 3.6 – Программный код выбора муравьем следующего города.

```
1  int choose_next_city(double *get_way_length, ant_t *ant, matrix_int_t *
       paths,
2  matrix_double_t *pheromons, matrix_double_t *attractiveness,
3  int index_last_city, meta_t meta_data_colony)
4  {
5      int count_cities = (*ant).visit_cities->size_array;
6      double denominator = 0, sum = 0;
7      int chosen_next_city;
8
9      for (int i = 0; i < count_cities; i++){
10         if (check_is_visited_city(i, ant->visit_cities, ant->
               count_visited_cities) == false)
11         {
12             sum = pow(pheromons->matrix[index_last_city][i],
                   meta_data_colony.alpha) +
13             pow(attractiveness->matrix[index_last_city][i],
                   meta_data_colony.beta);
14             denominator += sum;
15         }
16     }
17
18     double *probability = new double[count_cities]{};
19     for (int i = 0; i < count_cities; i++)
20     {
21         if (check_is_visited_city(i, ant->visit_cities, ant->
               count_visited_cities) == false)
22         {
23             sum = pow(pheromons->matrix[index_last_city][i],
                   meta_data_colony.alpha) +
24             pow(attractiveness->matrix[index_last_city][i],
                   meta_data_colony.beta);
25             probability[i] = sum / denominator;
26         }
27     }
28
29     chosen_next_city = get_value_from_fortuna_wheel(probability,
           count_cities);
30
31     *get_way_length = paths->matrix[index_last_city][chosen_next_city];
32     return chosen_next_city;
33 }
```

Листинг 3.7 – Программный код выбора города случайным образом ("колесо фортуны").

```
int get_value_from_fortuna_wheel(double *probability, int count_cities)
{
    srand(static_cast<unsigned int>(time(0)));
    double generated_probability = (double)(rand())/RAND_MAX;

    double sum = 0;
    bool is_index_city_find = false;
    int find_index = 0;
    for (int i = 0; i < count_cities && is_index_city_find == false; i
       ++){
        sum += probability[i];
        if (sum < generated_probability){
            find_index++;
        }
        else{
            find_index = i;
            is_index_city_find = true;
        }
    }

    return find_index;

}
```

3.5 Тестирование

Для тестирования используется метод черного ящика. В данном разделе приведена таблица 3.1, в которой указаны тесты:

Таблица 3.1 – Таблица тестов

№	Матрица	Ожидаемый результат
1	$\begin{pmatrix} 0 & 5 & 4 & 8 & 3 \\ 5 & 0 & 6 & 7 & 9 \\ 4 & 6 & 0 & 5 & 5 \\ 8 & 7 & 5 & 0 & 4 \\ 3 & 9 & 5 & 4 & 0 \end{pmatrix}$	30
2	$\begin{pmatrix} 0 & 6 & 8 & 1 \\ 6 & 0 & 3 & 5 \\ 8 & 3 & 0 & 9 \\ 1 & 5 & 9 & 0 \end{pmatrix}$	17

Для метода "грубой силы"и муравьиного алгоритма данные тесты пройдены успешно.

3.6 Вывод

В данном разделе был выбран язык программирования, среда разработки, приведены листинги модулей. Было успешно проведено их тестирование методом черного ящика по таблице 3.1.

4 Исследовательский раздел

В данном разделе приведены технические характеристики устройства; классы данных, на которых были проведены эксперименты; результаты параметризации и выборка из нее наилучших результатов.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- операционная система: Ubuntu 21.10;
- память: 8 GiB;
- процессор: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz.

Тестирование проводилось на ноутбуке, который был подключен к сети питания. Во время проведения тестирования ноутбук был нагружен только встроенными приложениями окружения, самим окружением и системой тестирования.

4.2 Постановка эксперимента

В муравьином алгоритме вычисления производятся на основе настраиваемых параметров. Рассмотрим два класса данных и выполним подбор параметров, при которых метод даст точный результат.

Будут рассматриваться матрицы размерности 10×10. Это необходимо для того, чтобы результат зависел от параметров.

В качестве первого класса данных будет является матрица стоимостей, в которой значения варьируются в диапазоне $[5;\ 10]$; во втором классе - значения лежат в диапазоне $[50;\ 100]$.

Будем запускать муравьиный алгоритм для всех значений α, β, $\rho \in [0, 1]$ с шагом $= 0.1$, пока не будет найдено точное значение или приближенное для каждого набора.

В результате тестирования будет выведена таблица со значениями $\alpha, \beta, \rho, , ,$ где — оптимальная длина пути, найденная муравьиным алгоритмом, - Разность между полученным значением и настоящей оптимальной длиной пути, а α, β, ρ — настраиваемые параметры.

4.2.1 Класс данных №1

Класс данных №1 описан следующей матрицей стоимостей (4.1):

$$M = \begin{pmatrix} 0 & 3 & 5 & 9 & 10 & 8 & 7 & 6 & 4 & 5 \\ 3 & 0 & 6 & 8 & 9 & 5 & 9 & 8 & 7 & 7 \\ 5 & 6 & 0 & 5 & 10 & 10 & 8 & 6 & 4 & 7 \\ 9 & 8 & 5 & 0 & 3 & 3 & 4 & 5 & 5 & 6 \\ 10 & 9 & 10 & 3 & 0 & 8 & 9 & 7 & 10 & 5 \\ 8 & 5 & 10 & 3 & 8 & 0 & 10 & 8 & 3 & 7 \\ 7 & 9 & 8 & 4 & 9 & 10 & 0 & 5 & 6 & 9 \\ 6 & 8 & 6 & 5 & 7 & 8 & 5 & 0 & 8 & 8 \\ 4 & 7 & 4 & 5 & 10 & 3 & 6 & 8 & 0 & 3 \\ 5 & 7 & 7 & 6 & 5 & 7 & 9 & 8 & 3 & 0 \end{pmatrix} \tag{4.1}$$

Результаты работы алгоритма с различными комбинация параметров представлены в приложении А. Таблица 4.1 содержит выборку параметров, которые помогают решить задачу наилучшим образом.

Таблица 4.1 – Выборка параметров

α	β	ρ	Длина	Разница
0.1	0.9	0.8	42	0
0.2	0.8	0.7	42	0
0.3	0.7	0.9	42	0
0.3	0.7	1	42	0
0.4	0.6	0	42	0
0.8	0.2	0.5	42	0
0.9	0.1	0.6	42	0
0.7	0.3	0.9	42	0

4.2.2 Класс данных №2

Класс данных №1 описан следующей матрицей стоимостей (4.2):

$$M = \begin{pmatrix} 0 & 53 & 65 & 79 & 90 & 82 & 76 & 61 & 47 & 55 \\ 53 & 0 & 62 & 84 & 95 & 53 & 91 & 68 & 77 & 76 \\ 65 & 62 & 0 & 50 & 99 & 73 & 58 & 61 & 64 & 57 \\ 79 & 84 & 50 & 0 & 93 & 63 & 54 & 55 & 65 & 86 \\ 90 & 95 & 99 & 93 & 0 & 98 & 92 & 67 & 73 & 54 \\ 82 & 53 & 73 & 63 & 98 & 0 & 82 & 81 & 73 & 70 \\ 76 & 91 & 58 & 54 & 92 & 82 & 0 & 50 & 60 & 90 \\ 61 & 68 & 61 & 55 & 67 & 81 & 50 & 0 & 81 & 82 \\ 47 & 77 & 64 & 65 & 73 & 73 & 60 & 81 & 0 & 63 \\ 55 & 76 & 57 & 86 & 70 & 70 & 90 & 82 & 63 & 0 \end{pmatrix} \tag{4.2}$$

Результаты работы алгоритма с различными комбинация параметров представлены в приложении А. Таблица 4.2 содержит выборку параметров, которые помогают решить задачу наилучшим образом.

Таблица 4.2 – Выборка параметров

α	β	ρ	Длина	Разница
0.1	0.9	0.6	554	0
0.2	0.8	0.2	554	0
0.3	0.7	0.4	554	0
0.4	0.6	0.7	554	0
0.4	0.6	0.9	554	0
0.8	0.2	0.2	554	0
0.9	0.1	0.5	554	0
0.6	0.4	0.8	554	0

4.3 Вывод

Была выполнена параметризация алгоритма муравьев для двух классов данных. Данная оптимизация при наборе параметров $\alpha = 0.3, \beta =$

$0.7, \rho = 0.4$ показала увеличение скорости выполнения в 400 раз по сравнению с полным перебором.

Заключение

В данной лабораторной работе были рассмотрены основополагающие материалы, которые потребовались для решения задачи коммивояжера. Были разработаны классы эквивалентности, проведена для них параметризация алгоритма муравьев. Был проведен сравнительный анализ двух методов решения задачи: полный перебор и муравьиного.

Муравьиный алгоритм позволяет обеспечить качественное решение, насколько это возможно. Для этого необходимо подбирать параметры для разных входных данных. Данный метод позволяет решить задачу коммивояжера в 400 раз быстрее полного перебора, что позволяет использовать его в качестве оптимизации. Алгоритм "грубой силы"является универсальным, для него не нужно проводить параметризацию, выдает гарантированно точный результат, но за длительное время (на размерностях матрицы, больше 20, вычисления могут занимать несколько дней, столетий). Поэтому важным критерием выбора способа решения является скорость выполнения: если она - ключевой фактор, то необходимо использовать муравьиный алгоритм.

Список литературы

[1] Штовба С.Д. "Муравьиные алгоритмы". в:Exponenta Pro. Математика в приложениях (2003).

[2] М.В. Ульянов. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ по ред. В.С. Аролович. Физматлит, 2008.

[3] C++11. [электронный ресурс], режим доступа: https://en.cppreference.com/w/cpp/11 (дата обращения 19.10.2021)

[4] Documentation for Visual Studio Code. [электронный ресурс], режим доступа: https://code.visualstudio.com/docs (дата обращения: 19.10.2021)

[5] Data and time utilities. [электронный ресурс], режим доступа: https://en.cppreference.com/w/cpp/chrono (дата обращения 20.10.2021)

Приложение А

Таблица 4.3 – Таблица коэффициентов для класса данных №1

α	β	ρ	Рез.	Погр.
0	1	0	42	0
0	1	0.1	46	4
0	1	0.2	45	3
0	1	0.3	43	1
0	1	0.4	42	0
0	1	0.5	42	0
0	1	0.6	42	0
0	1	0.7	42	0
0	1	0.8	44	2
0	1	0.9	42	0
0	1	1	42	0
0.1	0.9	0	42	0
0.1	0.9	0.1	43	1
0.1	0.9	0.2	42	0
0.1	0.9	0.3	45	3
0.1	0.9	0.4	44	2
0.1	0.9	0.5	42	0
0.1	0.9	0.6	42	0
0.1	0.9	0.7	43	1
0.1	0.9	0.8	42	0
0.1	0.9	0.9	42	0
0.1	0.9	1	42	0
0.2	0.8	0	42	0
0.2	0.8	0.1	42	0
0.2	0.8	0.2	43	1
0.2	0.8	0.3	42	0
0.2	0.8	0.4	44	2
0.2	0.8	0.5	42	0
0.2	0.8	0.6	42	0
0.2	0.8	0.7	42	0
0.2	0.8	0.8	45	3
0.2	0.8	0.9	42	0
0.2	0.8	1	42	0
0.3	0.7	0	42	0
0.3	0.7	0.1	43	1
0.3	0.7	0.2	42	0
0.3	0.7	0.3	43	1
0.3	0.7	0.4	42	0
0.3	0.7	0.5	43	1
0.3	0.7	0.6	42	0
0.3	0.7	0.7	43	1
0.3	0.7	0.8	42	0
0.3	0.7	0.9	42	0
0.3	0.7	1	42	0
0.4	0.6	0	42	0
0.4	0.6	0.1	44	2
0.4	0.6	0.2	43	1
0.4	0.6	0.3	42	0
0.4	0.6	0.4	42	0
0.4	0.6	0.5	44	2
0.4	0.6	0.6	42	0
0.4	0.6	0.7	42	0
0.4	0.6	0.8	42	0
0.4	0.6	0.9	42	0
0.4	0.6	1	42	0
0.5	0.5	0	42	0
0.5	0.5	0.1	42	0
0.5	0.5	0.2	43	1
0.5	0.5	0.3	43	1
0.5	0.5	0.4	42	0
0.5	0.5	0.5	42	0
0.5	0.5	0.6	42	0
0.5	0.5	0.7	42	0
0.5	0.5	0.8	42	0
0.5	0.5	0.9	42	0
0.5	0.5	1	42	0

α	β	ρ	Рез.	Погр.
0.7	0.3	0.4	42	0
0.7	0.3	0.5	43	1
0.7	0.3	0.6	46	4
0.7	0.3	0.7	42	0
0.7	0.3	0.8	44	2
0.7	0.3	0.9	42	0
0.7	0.3	1	42	0
0.8	0.2	0	42	0
0.8	0.2	0.1	42	0
0.8	0.2	0.2	43	1
0.8	0.2	0.2	42	0
0.8	0.2	0.3	42	0
0.8	0.2	0.4	46	4
0.8	0.2	0.5	42	0
0.8	0.2	0.6	42	0
0.8	0.2	0.7	42	0
0.8	0.2	0.8	45	3
0.8	0.2	0.9	42	0
0.8	0.2	1	42	0
0.9	0.2	0	42	0
0.9	0.2	0.1	42	0
0.9	0.2	0.2	43	1
0.9	0.2	0.2	42	0
0.9	0.2	0.3	42	0
0.9	0.2	0.4	43	1
0.9	0.2	0.5	42	0
0.9	0.2	0.6	42	0
0.9	0.2	0.7	42	0
0.9	0.2	0.8	42	0
0.9	0.2	0.9	42	0
0.9	0.2	1	42	0
1	0	0	42	0
1	0	0.1	42	0
1	0	0.2	42	0
1	0	0.3	43	1
1	0	0.4	42	0
1	0	0.5	42	0
1	0	0.6	42	0
1	0	0.7	42	0
1	0	0.8	42	0
1	0	0.9	42	0
1	0	1	42	0

Приложение Б

Таблица 4.4 – Таблица коэффициентов для класса данных №2

α	β	ρ	Рез.	Погр.
0	1	0	554	0
0	1	0.1	569	15
0	1	0.2	558	4
0	1	0.3	560	6
0	1	0.4	554	0
0	1	0.5	554	0
0	1	0.6	554	0
0	1	0.7	554	0
0	1	0.8	569	15
0	1	0.9	554	0
0	1	1	554	0
0.1	0.9	0	554	0
0.1	0.9	0.1	557	3
0.1	0.9	0.2	554	0
0.1	0.9	0.3	558	4
0.1	0.9	0.4	560	6
0.1	0.9	0.5	554	0
0.1	0.9	0.6	554	0
0.1	0.9	0.7	554	0
0.1	0.9	0.8	554	0
0.1	0.9	0.9	554	0
0.1	0.9	1	554	0
0.2	0.8	0	554	0
0.2	0.8	0.1	564	10
0.2	0.8	0.2	554	0
0.2	0.8	0.3	566	12
0.2	0.8	0.4	554	0
0.2	0.8	0.5	554	0
0.2	0.8	0.6	554	0
0.2	0.8	0.7	554	0
0.2	0.8	0.8	563	9
0.2	0.8	0.9	554	0
0.2	0.8	1	554	0
0.3	0.7	0	557	3
0.3	0.7	0.1	554	0
0.3	0.7	0.2	554	0
0.3	0.7	0.3	585	27
0.3	0.7	0.4	554	0
0.3	0.7	0.5	570	16
0.3	0.7	0.6	554	0
0.3	0.7	0.7	554	0
0.3	0.7	0.8	554	0
0.3	0.7	0.9	554	0
0.3	0.7	1	554	0
0.4	0.6	0	584	30
0.4	0.6	0.1	562	8
0.4	0.6	0.2	554	0
0.4	0.6	0.3	554	0
0.4	0.6	0.4	559	5
0.4	0.6	0.5	554	0
0.4	0.6	0.6	554	0
0.4	0.6	0.7	554	0
0.4	0.6	0.8	554	0
0.4	0.6	0.9	554	0
0.4	0.6	1	554	0
0.5	0.5	0	554	0
0.5	0.5	0.1	560	6
0.5	0.5	0.2	561	7
0.5	0.5	0.3	554	0
0.5	0.5	0.4	554	0
0.5	0.5	0.5	554	0
0.5	0.5	0.6	554	0
0.5	0.5	0.7	554	0
0.5	0.5	0.8	554	0
0.5	0.5	0.9	554	0
0.5	0.5	1	554	0
0.6	0.4	0	554	0
0.6	0.4	0.1	554	0
0.6	0.4	0.2	566	12
0.6	0.4	0.3	554	0
0.6	0.4	0.4	565	11
0.6	0.4	0.5	554	0
0.6	0.4	0.6	562	8
0.6	0.4	0.7	554	0
0.6	0.4	0.8	554	0
0.6	0.4	0.9	554	0
0.6	0.4	1	554	0
0.7	0.3	0	554	0
0.7	0.3	0.1	554	0
0.7	0.3	0.2	573	19

α	β	ρ	Рез.	Погр.
0.7	0.3	0.4	554	0
0.7	0.3	0.5	554	0
0.7	0.3	0.6	569	15
0.7	0.3	0.7	571	17
0.7	0.3	0.8	572	18
0.7	0.3	0.9	554	0
0.7	0.3	1	554	0
0.8	0.2	0	554	0
0.8	0.2	0.1	580	26
0.8	0.2	0.2	554	0
0.8	0.2	0.2	586	32
0.8	0.2	0.3	554	0
0.8	0.2	0.4	554	0
0.8	0.2	0.5	554	0
0.8	0.2	0.6	564	10
0.8	0.2	0.7	554	0
0.8	0.2	0.8	554	0
0.8	0.2	0.9	554	0
0.8	0.2	1	554	0
0.9	0.2	0	554	0
0.9	0.2	0.1	554	0
0.9	0.2	0.2	559	5
0.9	0.2	0.2	554	0
0.9	0.2	0.3	554	0
0.9	0.2	0.4	556	2
0.9	0.2	0.5	554	0
0.9	0.2	0.6	555	1
0.9	0.2	0.7	554	0
0.9	0.2	0.8	554	0
0.9	0.2	0.9	554	0
0.9	0.2	1	554	0
1	0	0	554	0
1	0	0.1	554	0
1	0	0.2	554	0
1	0	0.3	558	4
1	0	0.4	554	0
1	0	0.5	554	0
1	0	0.6	554	0
1	0	0.7	554	0
1	0	0.8	556	2
1	0	0.9	554	0
1	0	1	554	0