



Xi'an Jiaotong-Liverpool University

西交利物浦大學

Enhanced Ant-Colony Algorithm Based on Solving Trams-Route Problem

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science

Module Dissertation

Major Information and Computing Sci-
ence

Author Minglang Tuo

Teacher Dr. Jianjun Chen

Date 6th / June / 2020

Contents

Contents	ii
Abstract	iii
1 Declaration	1
2 Acknowledgements	2
3 Introduction	3
3.1 Introduction	3
4 Background	7
4.1 Mathematical Part	7
4.1.1 P versus NP Problem/ Travelling Salesman Problem	7
4.1.2 Coordinates of Points	8
4.2 Algorithm Part	9
4.2.1 A* Algorithm	9
4.2.2 Ant-Colony-optimize Algorithm	12
4.3 Programming Technology	14
4.3.1 OpenGL	14
4.3.2 GGplot2	14
5 Design	15
5.1 Design Methodology	15
5.2 Mathematical Model	16
5.3 Data Structure	17
5.4 Problem Analyze	17
5.5 Algorithm Design	19
5.5.1 Traffic Flow Enhancement Algorithm	19
5.5.2 Ant-Colony Algorithm	21
5.5.3 Cut-lines Algorithm	21
5.6 OpenGL Visualization	23
5.7 Data analysis	25

6	Implementation	27
6.1	Algorithm Implementation	27
6.1.1	Parameter Adjustment Class	28
6.1.2	Random data Input	29
6.1.3	Enhanced ant colony algorithm	31
6.2	Visualization Implementation	39
7	Evaluation	43
7.1	Test Background	43
7.2	Data Analysis	43
7.3	Test Result	46
8	Learning Points	47
9	Professional Issues	48
10	Appendix	49

Abstract

Traffic congestion in Suzhou has been increasing in recent years. Thus it's beneficial to develop a replaceable transport system, such as trams, to promote urban development [1]. As well as the different tram stations, generating one of the main tasks to be solved by logistics companies: the design of the tram route to optimization. Recently, one valid solution was proposed by several commercials for route planning. However, the cost of these applications is crucially high because of the multi-variable optimization including the requirement of passengers, the distance of stations and the maximum capacity of tram which makes the acquisition of application tough to afford [2]. In this paper, an optimize route design algorithm is developed for those route deciders to design more efficient routes based on the requirements. In order to generate the final optimize routes, the algorithm allows users to enter information about distance stations, passenger requirements and the number of required routes. In this algorithm, the core problem can be regared as traveling salesman problem and one of the heuristic approachs named ant-colony algorithm has been improved based on a-star algorithm and utilized, which is the core idea of algorithm to optimize the routes. Furthermore, this reprot also discusses how the final solutions are influenced by the different ant-colony parameters based on different range of tram-stations and compares the effect between the original ant-colony algorithm and enhanced ant-colony algorithm.

Key Words: Combinatorial Optimization, Tram-Route Problme, Traveling Salesman Problem, Ant-Colony Algorithm, A-Star Algorithm, Enhanced Ant-Colony Algorithm.

Chapter 1

Declaration

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,
Minglang Tuo

Chapter 2

Acknowledgements

Firstly, thanks for Dr. Chen's guidance, support and encourage when I met the difficult problem and the wrong direction during the project. In each essential step, he patiently provides the core information, which make me able to achieve the milestone in the development of the project. Working under his guidance equipped me with the necessary skills to make effective contributions to the research team, also laid the foundation for me to enter the next stage in my career. Secondly, I would like to thank my real life friends and online friends in stackExchange, who help to handle my subtle programming problems and some mathematical problems in the development process. Thirdly, I appreciate my parents and Xi'an Jiaotong-Liverpool University, which helps me to develop the confident and key skills needed in my career path.

Chapter 3

Introduction

3.1 Introduction

The introduction part reviews the relevant problem and literature to solve the tram route problem. In the modern society, people pay more and more attention to the economy and efficiency, hence the combinatorial optimization becomes more and more essential. The initial combinatorial optimization was proposed by the Kantorovich and Koopmans based on the coherent mathematical discipline. However, the theory is stuck in a bottleneck until other important theory developed, such as the matching and matroids had been achieved by Edmond, Cook and Karp's theory of NP-completeness had been proved. The most classical optimization problems in the early days contain six types, including assignment, transportation, maximum flow, shortest tree, shortest path, and the traveling salesman problem [3]. Nowadays, optimization problems are constantly involved in our daily life. Not only the primitive animals select the optimal nest or the social animals to develop suitable hunting strategies, but also the doctors make schedule about the most reasonable time of patients or the tour guide needs to organize the vacation route for tourist attractions.

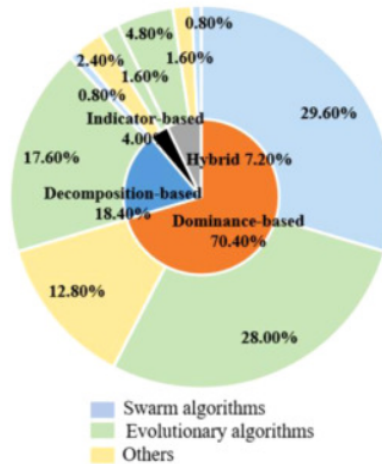


Fig. 3.1: Multi-Objective Metaheuristics for MODOPs are Classified

Different from the previous combinatorial optimization problem, the present optimization problem has been developed into multi-objective discrete optimization problems (MODOPs), which can be expressed by mathematical formula to satisfy the optimal choice. After that, academic and industrial concentrate on the multi-objective problems because their universal applications in reality [4]. There are lots of multi-objective optimization approaches have been cerated that can be seperated into two classifications, including classical method and metaheuristics [5]. Classical multi-objective optimization methods contain scalarization-based methods, such as the weighted sum method,goal programming, weighted chebyshev method [6], the two-phase method [7], and the multi-objective branch and bound approaches [8]. Contrasting with classical approaches, multi-objective metaheuristics have been more universally utilized to accomplish realistic multi-objective problems in the existing multi-objective optimization literature. The reasons are their capacity of providing great heuristic optimum-seeking abaility and generating eeffective Pareto optimal solutions within a reasonable computation time. These two graphs are established on data from all literature from 2008 to 2019 in 39 journals. Fig.3.1 shows that the multi-objective metaheuristics for MODOPs are categorized into four contents based on for disparate selection mechanisms, including dominanace-based, decompostion-based, indicator-based and hybride selection mechanism-based metaheuristics. As for the first three selection mechanisms, 70.40% of machinery is dominance-based, 18.40% of machinery is decompositon based, and 4.00% is other selection. Swarm algorithms, evolutionary algorithms, and others, are purpose to search techniques to seek better solutions in multi-objective selection mechanisms.

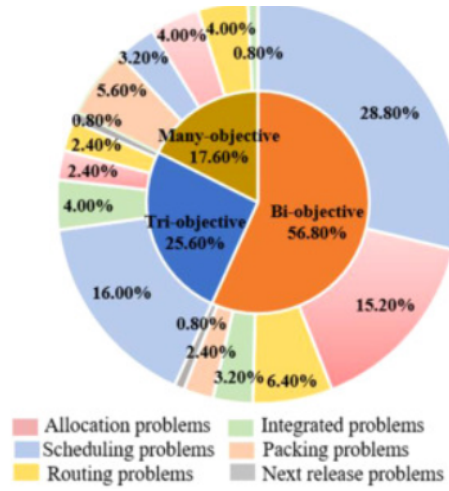


Fig. 3.2: The problems Based on MODOPS

And for the different problems in reality, the data can be observed on the Fig.3.2. In addition, the previous lituratures are concerned about MODOPs with two or three objectives primarily, which account for 56.80% and 25.60% respectively. Scheduling, allocation, routing, packing, and their integrated problems are settled down by the different type of

MOOPs. The tram-route problem is one of the allocation problems based on Traveling Salesman Problem whose target is how to optimally allocating the passengers and set the different routes rely on the requirements of passengers.

Multi-objective metaheuristics		Number of references	Percentage
Selection mechanisms	Search methods		
Dominance-based	Genetic algorithm (GA) [26]	25	20.00%
	Particle swarm optimization (PSO) [27]	13	10.40%
	Artificial bee colony (ABC) [28]	5	4.00%
	Differential evolution (DE) [29]	4	3.20%
	Grey wolf optimization (GWO) [30]	4	3.20%
	Memetic algorithm (MA) [31]	4	3.20%
	Local search (LS) [32]	4	3.20%
	Harmony search (HS) [33]	4	3.20%
	Ant colony optimization (ACO) [34]	2	1.60%
	Electromagnetism mechanism algorithm (EMA) [35]	2	1.60%
	Other approaches (e.g. FA ^a , AFSA ^b , BFOA ^c , CCA ^d , WWO ^e , SFLA ^f , IWO ^g)	21	16.80%
	Sub-total	88	70.40%

Fig. 3.3: The Table for Different Dominance-Based Multi-Objective Metaheuristics

The ACO is a dominance-based multi-objective metaheuristic, which comprises Pareto dominance [9], ε -dominance [10] and Lorenz dominance [11], among which Pareto dominance is the most commonly used. Although there are other dominance-based multi-objective metaheuristics (Fig.3.3) like tabu search algorithm which designs a tabu table like human brain to optimize the solution step and step, the genetic algorithm is a good choice to solve the problem, which models to the biological evolution process of genetic mechanism to solve the computational problem. The features for three types of algorithm are different and each approach is appropriate for different problems. The ant-colony algorithm has relatively high efficient then others when the dataset is not mass. In consequence, the ultimate decision for the project is the ant-colony-optimization as the solution for the problem in the project. In consequence, they are appropriate for different problems. As for the allocating passengers, it's nessessary for construct the practical path-finding method. Afterwards, the a-star is optimal path-finding method to modifies and enhances ant-colony algorithm. The matrix (two-dimensional array) and graph as the data structure process the information.

In this project, we hope to provide an application for tram company decision makers to set tram routes. It allows to input distance matrix (distance between stations), the

passenger demand matrix which represent the requirement of passengers and the number of tram lines. After the optimization of three modified algorithms, including the traffic flow enhanced algorithm, the traditional ant colony algorithm and the line cutting algorithm, the lines will be the result as output. Later on, we will figure out the optimal parameter of the enhanced ant colony algorithm and the traditional ant colony algorithm for a certain range of input (for example, when the station interval is between 40 and 60), and compare the transport efficiency between them through a certain data set. Finally, based on the data of experiment, it concludes that the enhanced ant colony algorithm is indeed more efficient than the traditional ant colony algorithm.

This paper is divided into several parts, Chapter 4 mainly introduces the related problems of mathematical knowledge and software knowledge, as well as the technology used in this application. Chapter 5 points out the design of the project. Chapter 6 describes the implementation of this project. In chapter 7, the experiment of the algorithm in virtual data is given, and the results are analyzed. Chapter 8 and chapter 9 respectively mentions the differences between the learning points and professional issues of the program.

Chapter 4

Background

4.1 Mathematical Part

This part will talk about the mathematical background of the project, including the traveling salesman problem that belongs to NP problem and the transformation from the distance matrix to the points of cartesian coordinate system by the eigenvalue decomposition.

4.1.1 P versus NP Problem/ Travelling Salesman Problem

The P versus NP problem is one of the seven millennial problems which worth a huge prize in the clay institute of mathematics. In addition, It also the biggest problem in computer science. Polynomial Problem (P) means the complex problem which can be settled down in polynomial time [12]. The polynomial time can be observed as:

$$f(x) = bx^n + (b - 1)x^{n-1} + (b - 2)x^{n-2} + + x \quad (4.1)$$

In contrary, Nondeterministic Polynomial Problem (NP) means the complex problem cannot be determined to be solved in polynomial time. However, it can determine a guess solution is useful in polynomial time. And if we can demonstrate an algorithm to solve the NP problem in polynomial time, we can prove that $P = NP$. As for arranging reasonable tram stations lines, It's obviously impossible for the computer to calculate the optimize lines within a polynomial time. Therefore, the algorithm such as metaheuristics method will be introduced and give the solution by iteration.

The traveling salesman[13] problem is cited as a classic combinatorial optimization principle to solve the tram station problem. It describes a salesman who travels to several cities to sell his goods. It's essential to choose the route to make the entire journey shortest. The principle of arranging reasonable traffic paths is actually based on the problem because of similar situation. More specifically, the problem assume that tram goes through all the stations, and it needs to transport all the passengers to their desired destination. Actually, the only different is the criteria that changes to the time of passengers.

4.1.2 Coordinates of Points

During the project, It's necessary to transfer the collecting data to the real coordinates of points. As we know, the information of stations and the information about requirements of passengers can be stored in the matrix, like the following matrixs:

$$DistanceMatrix \begin{bmatrix} 0 & 5 & 8 \\ 5 & 0 & 2 \\ 8 & 2 & 0 \end{bmatrix}$$

The distance matrix named Euclidian distance matrix can represent the distance between each stations. For example, 5 meters is the distance from station1 to station2.

$$PassengerMatrix \begin{bmatrix} 0 & 1 & 2 \\ 5 & 0 & 2 \\ 8 & 3 & 0 \end{bmatrix}$$

The passenger matrix means the requirements for the passengers. For example, the element in row2/line1 means that there are 5 people who want to take tram from station2 to station1.

Subsequently the distance matrix can be converted into coordinates, and we can use the the following equation firstly[14]:

$$M_{ij} = \frac{D_{1j}^2 + D_{i1}^2 - D_{ij}^2}{2} \quad (4.2)$$

One thing that is good to know in case the dimensionality of the data that producted the distance matrix is not known is that the smallest (Euclidean) dimension in which the points can be embedded is given by the rank k of the matrix M. No embedding is possible if M is not positive semi-definite.

The coordinates of the points can now be obtained by eigenvalue decomposition: if we write $M = USU^T$, then the matrix $X = U\sqrt{S}$ (you can take the square root element by element) gives the positions of the points (each row corresponding to one point). Note that, if the data points can be embedded in k-dimensional space, only k columns of X will be non-zero (corresponding to k non-zero eigenvalues of M). The principle can be explained in the following steps:

1. If D comes from distances between points, then there are $\mathbf{x}_i \in \mathbb{R}^m$ such that:

$$D_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j)^2 = \mathbf{x}_i^2 + \mathbf{x}_j^2 - 2\mathbf{x}_i \cdot \mathbf{x}_j. \quad (4.3)$$

2. Then the matrix defined above takes on a particularly nice form:

$$M_{ij} = (\mathbf{x}_i - \mathbf{x}_1) \cdot (\mathbf{x}_j - \mathbf{x}_1) \equiv \sum_{a=1}^m \tilde{x}_{ia} \tilde{x}_{ja}, \quad (4.4)$$

where the elements $\tilde{x}_{ia} = x_{ia} - x_{1a}$ can be assembled into an $n \times m$ matrix \tilde{X} . In matrix form,

$$M = \tilde{X} \tilde{X}^T. \quad (4.5)$$

Such a matrix is called a Gram matrix. Since the original vectors were given in m dimensions, the rank of M is at most m (assuming m is greater than n).

3. The points we get by the eigenvalue decomposition described above need not exactly match the points that were put into the calculation of the distance matrix. However, they can be obtained from them by a rotation and a translation. This can be proved for example by doing a singular value decomposition of \tilde{X} , and showing that if $\tilde{X} \tilde{X}^T = X X^T$ (where X can be obtained from the eigenvalue decomposition, as above, $X = U \sqrt{S}$), then X must be the same as \tilde{X} up to an orthogonal transformation.

4.2 Algorithm Part

The knowledge of the algorithm in the project will be introduced in this part. There exists two main heuristic methods to solve the problem. One heuristic method is about finding the shortest path, commonly known as a-star algorithm. Another heuristic method named ant-colony-optimize algorithm is the core algorithm to construct different routes as the final result.

4.2.1 A* Algorithm

The A* algorithm as a new heuristic method for finding the path which is completely different from the uniform-cost search algorithm like dijkstra algorithm. It can continuously estimate the distance from the initial point to the end point in the tree or graph in each step. Then, the best optimize path will be selected during the each step. Hart et al. (1968) extended the Dijkstra algorithm (Dijkstra, 1959) to propose the A* algorithm[15]. The A* algorithm was used in lots of dimensions. For example, game units in Starcraft go to find the path, which is formally done using the A* algorithm over and over again, allowing units to find the path to the target you mouse clicked on. The core principle of A* algorithm can be shown in the following formula:

$$F(n) = G(n) + H(n) \quad (4.6)$$

$G(n)$ means the cost that the path have been already walked. And $H(n)$ using the greedy method to estimate and choose the path which has cheapest cost from the current location

to the dest location. $F(n)$ is the complete cost for the path. Comparing to the dijkstra algorithm, the A* algorithm just adds new parameter $H(n)$ to the algorithm.

The pseudocode of A* algorithm can be observed as following:

- Input: A graph G which contains lots of nodes. The start node for searching (node_start). The dest node for finding (node_goal).
- Output: A shourtest route from node_start to node_goal.
- Notes: Two lists as data sturcture: OPEN and CLOSE.
- OPEN consists on unvisited nodes or the modes haven't extended. This is the list of pending jobs.
- CLOSE consists on visited nodes that have and expanded

```

1 Put node_start in the OPEN list with  $f(\text{node\_start}) = h(\text{node\_start})$  (initialization)
2 while the OPEN list is not empty {
3   Take from the open list the node node_current with the lowest
4    $f(\text{node\_current}) = g(\text{node\_current}) + h(\text{node\_current})$ 
5   if node_current is node_goal we have found the solution; break
6   Generate each state node_successor that come after node_current
7   for each node_successor of node_current {
8     Set successor_current_cost =  $g(\text{node\_current}) + w(\text{node\_current}, \text{node\_successor})$ 
9     if node_successor is in the OPEN list {
10      if  $g(\text{node\_successor}) \leq \text{successor\_current\_cost}$  continue (to line 20)
11    } else if node_successor is in the CLOSED list {
12      if  $g(\text{node\_successor}) \leq \text{successor\_current\_cost}$  continue (to line 20)
13      Move node_successor from the CLOSED list to the OPEN list
14    } else {
15      Add node_successor to the OPEN list
16      Set  $h(\text{node\_successor})$  to be the heuristic distance to node_goal
17    }
18    Set  $g(\text{node\_successor}) = \text{successor\_current\_cost}$ 
19    Set the parent of node_successor to node_current
20  }
21  Add node_current to the CLOSED list
22 }
23 if (node_current != node_goal) exit with error (the OPEN list is empty)

```

Fig. 4.1: Pesudo Code for A* Algorithm

There flow-chart of A* algorithm:

1. The A* algorithm has two diffent Lists and both of them are empty. One list stores the visited or expanded nodes(CL), another list reserves the unvisited nodes(OL).
2. An destination-node B is indexed to the OL.
3. Randomly choose the node as start node as B in the OL when value of f in all nodes are zero. Subsequently, it can be regarded as the current node C, then remove it from the OL and add it to the CL.

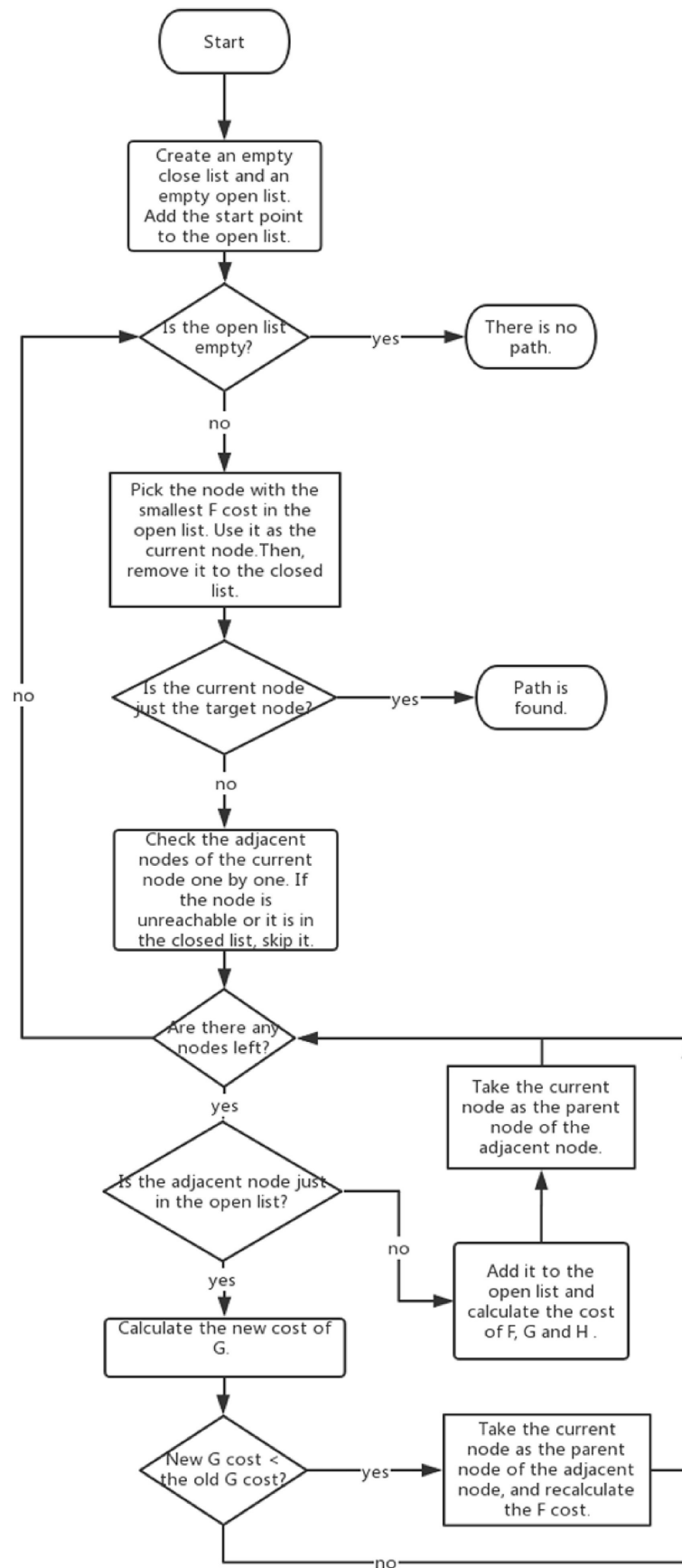


Fig. 4.2: Flow Chart for A* Algorithm

4. Observe the adjoining nodes of the current node C. In the meanwhile, ignore the node that is unreachable or already in the CL.
 - a. Add the observed nodes to the OL and calculate the cost of F, G, and H. More specifically, We always add the minimize value of adjacent nodes to the OL and estimate whether the value of old nodes is smaller then new path to the old nodes. If yes, we develop the new path.
5. Loop Steps (3) and (4) until the target point B is added to the OL, which indicates that the path is found OR the O always in the List(CL) that means no path to the B.

Ultimately, move along the parent node of each point from the target point O until the Random Rstarting point B is added to the path. Save the path. The flowchart of the A* algorithm is shown in Fig.4.2.

4.2.2 Ant-Colony-optimize Algorithm

Algorithm 1 The framework of a basic ACO algorithm

input: An instance P of a CO problem model $\mathcal{P} = (\mathcal{S}, f, \Omega)$.
 InitializePheromoneValues(\mathcal{T})
 $s_{bs} \leftarrow \text{NULL}$
while termination conditions not met **do**
 $\mathcal{S}_{iter} \leftarrow \emptyset$
 for $j = 1, \dots, n_a$ **do**
 $s \leftarrow \text{ConstructSolution}(\mathcal{T})$
 if s is a valid solution **then**
 $s \leftarrow \text{LocalSearch}(s)$ {optional}
 if ($f(s) < f(s_{bs})$) or ($s_{bs} = \text{NULL}$) **then** $s_{bs} \leftarrow s$
 $\mathcal{S}_{iter} \leftarrow \mathcal{S}_{iter} \cup \{s\}$
 end if
 end for
 ApplyPheromoneUpdate($\mathcal{T}, \mathcal{S}_{iter}, s_{bs}$)
end while
output: The best-so-far solution s_{bs}

Fig. 4.3: Pesudo Code for Ant-Colony-optimize Algorithm

Another heuristic algorithm is ant-colony-optimize algorithm, which is based on the simulation of the ant-colony looking for food to find the optimal path. In the early 1990s, M. Dorigo and colleagues discover and create the ant colony optimization (ACO) as an original nature-inspired metaheuristic for solving hard combinatorial optimization (CO) problems [16]. According to the observation and research of entomologists, the ants in the biological world have the ability to find the shortest path from their nest to the food source without any visible hint. Meanwhile, they can change the shortest path depends on the change of environment, adaptively search for new path and generate new choices. The reason is the pheromones which can be regard as the special secretions in the paths. It causes that the ants are able to detect and influence their subsequent movements. For example, with ants passing along some paths more and more, the rate of pheromone on

these paths becomes more greater which leads to the subsequent ants more likely to choose the paths. The process of selection is known as ant autocatalytic behavior. The pseudo code for the algorithm can be observed in Fig.4.3.

There are two different formulations which apply in two different functions in pseudo code. One is the equation of RouteSelection corresponding the ConstructSolution(), another is the equation of PheromoneUpdate corresponding the ApplyPheromoneUpdate().

As for the formula of RouteSelection, it represents the probability for the ant k choose the next random station j from station i . It's like this:

$$p_{ij}^k = \frac{\tau_{ij}^\alpha + \eta_{ij}^\beta}{\sum_{j \in \Lambda} \tau_{ij}^\alpha + \eta_{ij}^\beta} \quad (4.7)$$

- m : the total number of ants.
- k : the ant is exploring in the stations, whose serial number is k . ($1 < k < m$)
- i : the current station for the ant k .
- j : the adjacent stations for the current station.
- τ_{ij} : the pheromone of edge that is from station i to station j .
- η_{ij} : the visibility of edge that is from station i to station j . In addition, the longer the road, the less visibility in the road.
- Λ : the station set which is adjacent to the current station.
- α : the parameter factor.
- β : the parameter factor.

As for the formula of PheromoneUpdate, the function is updating the pheromone in each iteration. It's like this:

$$\tau_{ij}(n+1) = \rho * \tau_{ij}(n) + \sum_{k=1}^m \Delta \tau_{ij}^k, \quad (4.8)$$

- n : the total times for m ants to walk through all the stations.
- $\tau_{ij}(n)$: the pheromone of edge between the station i and station j in the previous iteration.
- $\tau_{ij}(n+1)$: the pheromone of edge between the station i and station j in the current iteration.
- $\Delta \tau_{ij}^k$: after this iteration, the ant k remains the pheromone in the edge between the station i and station j .

- ρ : Evaporation rate ($0 < \rho < 1$)

There is an additional formula for calculating the $\Delta\tau_{ij}^k$:

$$\Delta\tau_{ij}^k = \frac{Q}{L_k} \quad (4.9)$$

- L_k : the total length for k ant to walk athrough all the stations in current iteration.
- Q : the parameter factor to adjust the pheromone increment in suitable range.

The detailed flow chart and analysis will be explained in the design section.

4.3 Programming Technology

The final part will discuss the programming technology in the project. There are three main technologies implementing in the project. Firstly, the core algorithm and framework are written by python. Secondly, the visualization of route is exposed by the OpenGL library in C++. Thirdly, the test and parameter adjustment of experiment are presented through the ggplot library of R language.

4.3.1 OpenGL

OpenGL (Open Graphics Library) is a cross-language and cross-platform application programming interface (API) for rendering 2D and 3D vector Graphics [17]. The interface consists of nearly 350 different function calls used to draw everything from simple graphics bits to complex three-dimensional scenes. The other programming interface system is Direct3D for Microsoft Windows only. OpenGL is often used in CAD, virtual reality, scientific visualization programs and video game development. The project will use the 2D vector graphic to visualize for the process of searching the routes.

4.3.2 GGplot2

As an R package from the tidyverse, ggplot2 has the powerful function to customizing graphs amd easily delete or modify components at a high level of abstraction. Its popularity is down to the simplicity of customizing graphs and removing or altering components in a plot at a high level of abstraction [18]. Distinct graph available in the ggplot as well as other popular libraries, such as matplotlib in python. The experiment data and results are analyzed by the library in the project.

Chapter 5

Design

5.1 Design Methodology

This development of this project adopts incremental model. The product is designed, implemented and tested incrementally, until the project is completed. The product consists of multiple components, each of which is designed and built separately. The increments refer to a series of releases, and each increment will provides more functionalities to users [19]. After the first increment, a version which can be used by the users is delivered. After that, the plan for next increment is developed based on user feedback. This process will continues until the complete product is delivered [20].

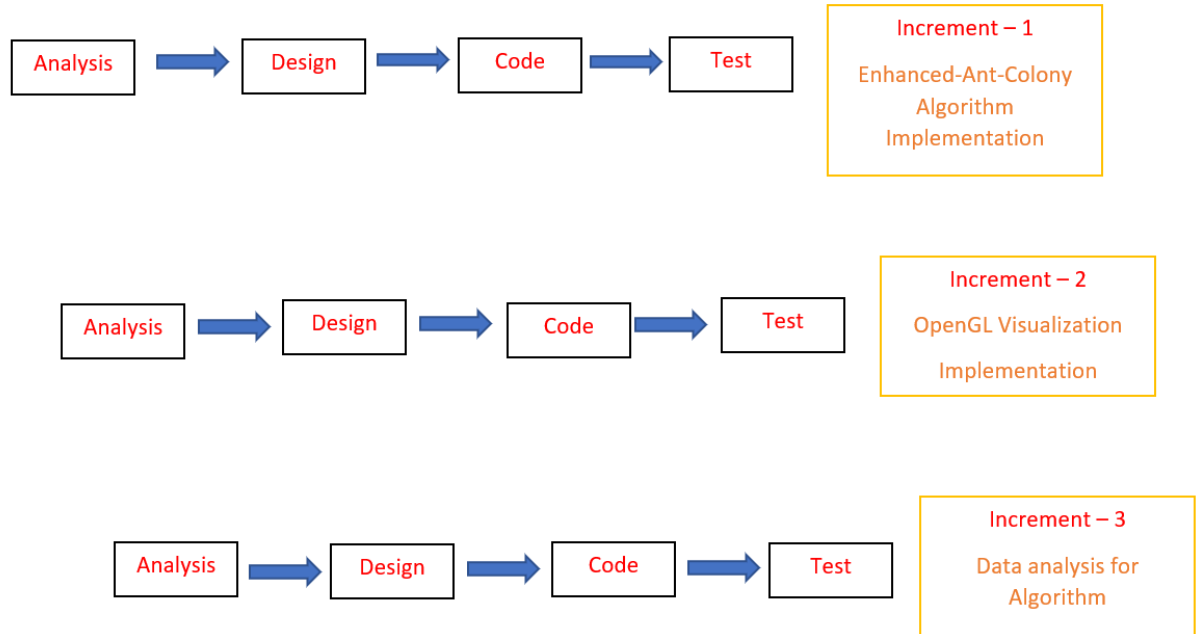


Fig. 5.1: Increment Model

The general process of increment model is shown in Fig.5.1. The Increment-1 of project is constructing Enhanced-Ant-Colony Algorithm which includes the Traffic-Flow-Enhance Algorithm, Ant-Colony Algorithm and Cut-lines Algorithm. Before implementing the

algorithm, it's vital to utilize the random method to generate some virtual data for verifying the algorithm. After the first increment delivered, the core function have been already created. Furthermore, the user can input some relevent value to simulation the virtual tram routes and the program will output in consolo. The Increment-2 is delivering an OpenGL visualization project. After the second increment delivered, the stations can be displayed in the cartesian coordinate system. Moreover, customers can observe the process of optimizing tram routes. Data analysis for algorithms is significance milestone for the Increment-3. After the third increment achieved, customers can analyze to get the adequate parameter for the algorithm established on different range of problem, such as 40 tram-stations, 100 passengers and 4 routes. Eventually, the experiment are going to compare the efficiency between the original ant-colony algorithm and the enhanced ant-colony algorithm through Some experiments under the same conditions.

5.2 Mathematical Model

The section talks about building the mathematical model. The section introduce the mathmatical model established on regulating the route of tram station in real world:

1. There exists some tram stations, such as $n_1, n_2, n_3, \dots, n_m$
2. There are the random number of passengers in each stations, such as $node_1$ containing $Cn_{11}, Cn_{12}, Cn_{13}, \dots, Cn_{1i}$.
3. Each passenger has their own destination, which can be expressed as the function: $f(p)$.
4. For the tram, the tram speed is fixed as V_t . For the passenger, the walk speed is fixed as V_w . The time for the passenger can be represented as $t(c)$.
5. The edge between each stations can be represnet as j , each edge connect with two different stations.
6. A tram routine visite the stations in order.
7. The tram routine will be divided into fixed different lines, such as L_1, L_2, L_3, \dots

In conclusion, there are several stations and random passenger in each station. The purpose of passenger is arriving his own destination. Then the design for the project is designing different routes to satisfy the benefits of passenger as much as possible. In addition, The standard for the routes is the total speed time in all the passengers. Consequently, the design needs to minimize the criteria.

5.3 Data Structure

In the project, there are one essential data structure for utilizing, which is directed graph. Besides, it can be illustrated by matrix(2-dimensional array). There are two functions for the graph. As for the input stream and output stream, two dimensional arrays stores these relevant information, including passenger information, station information, pheromone information and traffic-flow information. Also it can operate the matrix operation, such as eigenvalue decomposition in the project. Another is the directed graph used to construct the traffic network. The following algorithms, such as a-star algorithm, traffic flow enhancement algorithm, ant colony algorithm and cut-lines Algorithm carried out based on the graph. For example, when the a-star algorithm was executed, the matrix store the information, such as $G(n)$ and $H(n)$, and the generation of shortest route on the foundation of graph.

5.4 Problem Analyze

The section will introduce how to analyze the real problem and transfer the problem to mathematical model. After that, what's the most important is how to design the enhanced algorithm based on ant-colony algorithm to solve the mathematical model.

The description of heuristic method generates the feasible solution for the combinatorial optimization problem, which simulating the natural things. Although the solution frequently has the deviation from the optimal solution, it can keep optimized by iterations. In the paper, we use the improved algorithm based on ant-colony algorithm because the robustness is stronger than other heuristic algorithm and it's a parallel algorithm and positive feedback algorithm. Depend on its advantages, we can easily to solve the tram-route problem. However, it's also have some disadvantages including the slow of convergence speed, easily falling to local optimal and easily affected by parameters. Our main mission is improving the disadvantages and enhance the algorithm established on features.

The flow chart is shown in Fig.5.2. The program starts by inputting the information about the relevant station to the system, including the station information and passenger information. Then, the traffic flow enhancement algorithm handles the distance matrix and passenger information matrix firstly and generates the traffic flow matrix. The intention of the traffic flow matrix is placing the pheromone in advance, which refines the rate

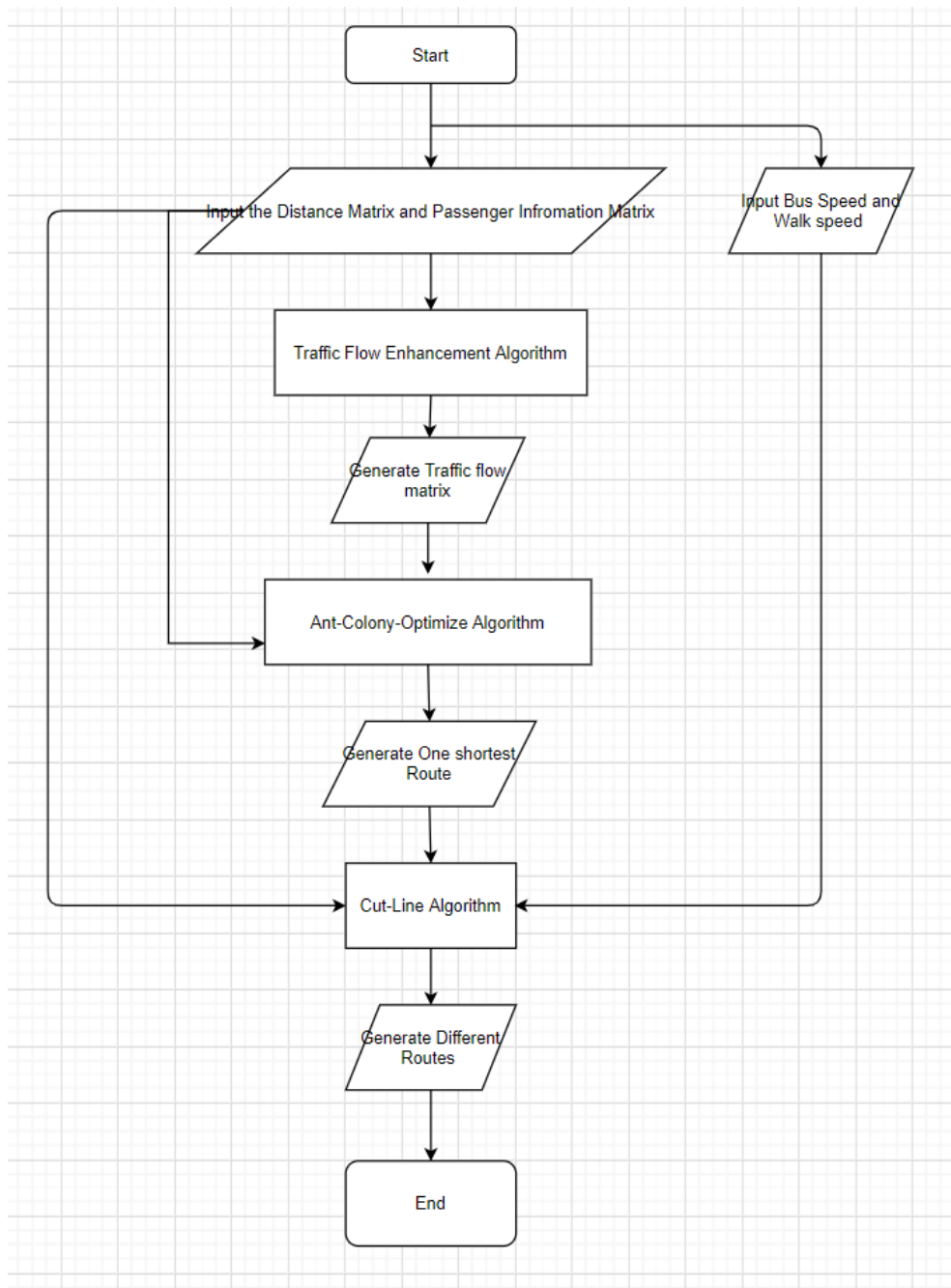


Fig. 5.2: The Process of Algorithm

of convergence in the ant-colony algorithm. After that, one shortest route will be developed by the ant-colony-optimize algorithm. Ultimately, based on the speeds of distinctive modes of transportation, the algorithm estimates the criteria(time) for each passengers to generate the inclusive criterion. Subsequently, the minimum chop way will be calculated.

5.5 Algorithm Design

Thirdly, the section will describe how we design the enhance algorithm based on the origin ant-colony algorithm. It can be separated three main parts. There are Traffic Flow Enhancement Algorithm, Ant-Colony Algorithm and Cut-lines Algorithm.

5.5.1 Traffic Flow Enhancement Algorithm

The traffic flow enhancement algorithm is modified according to the a-star algorithm and it's an iteration algorithm. The result is that we place the pheromones regard as food to the edges between different stations in advance. The beneficial for the processing is decreasing the speed of convergence when running the following ant-colony algorithm. The input for the algorithm are distance matrix and passenger Information matrix. Then, the output is the traffic flow matrix. The flow chart can be observed in Fig.5.3.

The specific processes can be displayed in the following:

1. Read the information in the passenger matrix and use the A* algorithm for each passengers.
2. For every passenger, base the formulate of A* algorithm, we choose the start station of passenger as start station S and the destination station of passenger as destination D . Firstly, we set the value of $G(S) = 0$, and $H(S) = \text{Distance Matrix}[S][D]$
3. After extension of start station S in the graph, we get other points of value in the matrix, such as $G(P_1) = \text{Distance matrix}[S][P_1]$, $H(P_1) = \text{Distance matrix}[P_1][D]$, and create a new variable (Current Station C)
4. Then, we compare whether the direct distance (value equals $F(C)$) from start point to the destination point with the all other points ($P_1, P_2, P_3...$). If answer is yse, then output the direct line. If answer is no, jump to next step.
5. We use the merge-sort to sort other stations $F(P_1) F(P_2) ,F(P_3)....$ and choose the minimum value as the next selected station. If there exists same value of $F(p)$ for different stations, we will look for the next extension of each stations who have same value of F . And comparing their value of F and selecting the station.

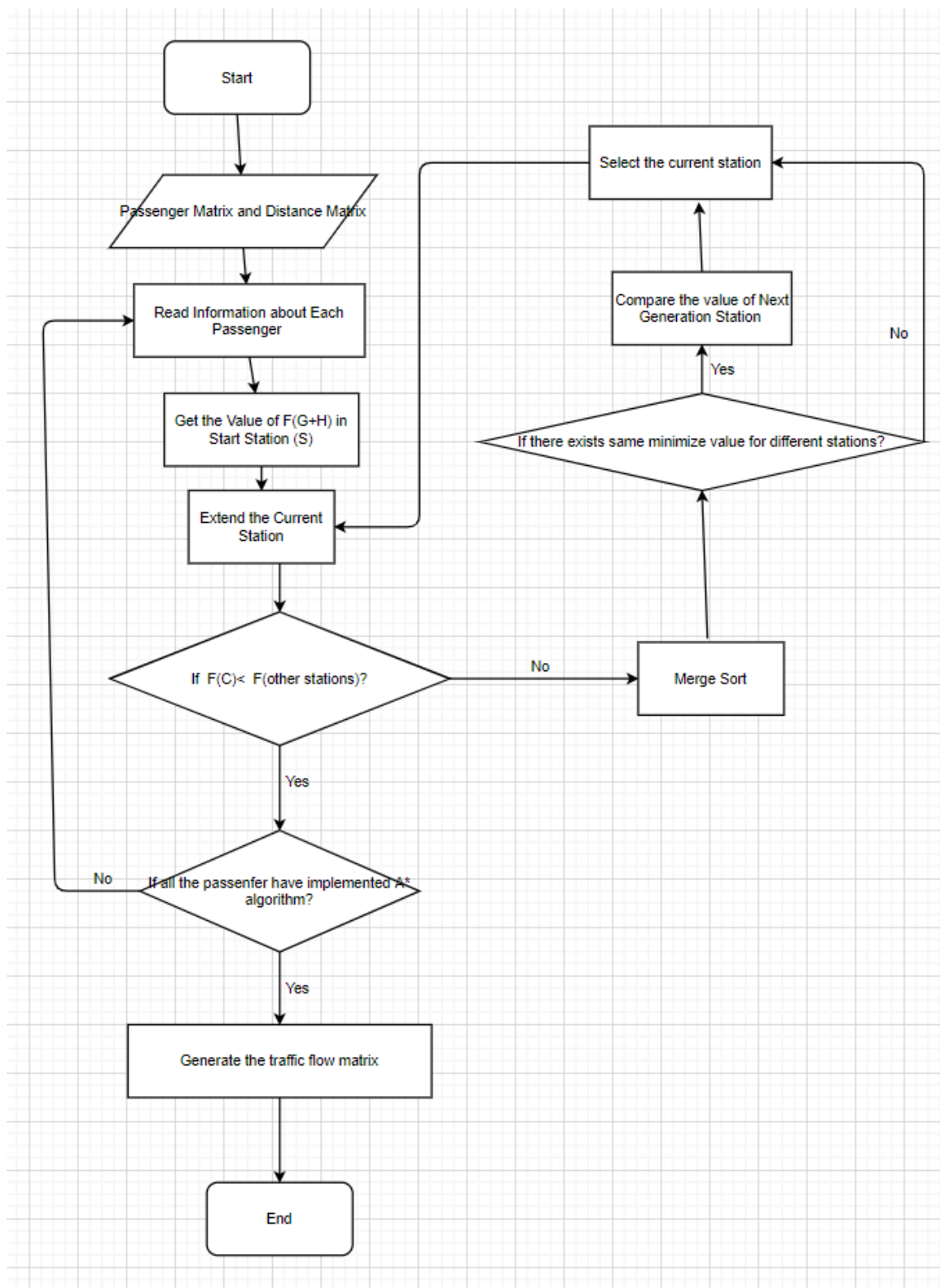


Fig. 5.3: The Flowchart about Traffic Flow Enhancement Algorithm

6. we can now regard the current station as the another start station and jump to step2. However, only change is that the value of G is cumulative, it will add all the length of the path traveled.
7. Until all the passenger have been implemented by A* algorithm and based on the routes create the traffic flow matrix, which records how busy each path.

The pseudo-code of the processing can be like this:

5.5.2 Ant-Colony Algorithm

Next, we implement the ant-colony algorithm to generate one route. For the original ant-colony algorithm, we initialize the stations in the map. And the fixed number of ants were randomly distribute in different stations. As for the ant, it has the tabu-table to store the traveling stations and prevents it to arrive the same station again at current iteration. Once all the ants finish the current iteration. The pheromone will be updated. Then, we choose the shortest route as temporary route in the iteration. After a fixed number of iteration, we choose the minimize route in all the iteration lines as the final result. The different for the algorithm to the original ant-colony algorithm is that we place the pheromone based on the traffic flow matrix in advance. The flow chart can be observed in the Fig.5.4.

The evaluation criterion for choosing the path is the time. And we know that the tram speed is V_t , and it's important for finding the shortest length for all the stations.

1. As for the part of Ants traverse station in the flowchart, the aim of ants is exploring all the possible cases for traversing all the stations. And we get the temperate shortest path by comparing the length of each ant walking. During the ant choose the station randomly, the formula of 4.7 will be used.
2. As for the iteration part, once finishing one iteration, all the pheromone will be updated by the formula of 4.8 and 4.9. And the two temperate shortest paths will compare with each other. The smaller one will be the new temperate shortest path. When all the iteration has finished, the final temperate shortest path will be the shortest path.

5.5.3 Cut-lines Algorithm

Finally, we design the line segment cutting algorithm to cut all the possible cases to get the optimal lines for transferring the passengers. The processing for the algorithm is simple, based on one shortest route in ant-colony algorithm, we cut the line which depends the cut-edge. Then we calculate the criteria for all the passengers in different cases.

The criteria can be represented like this: we assume every passenger run the route to the destination based on a-star algorithm and we caculate the spend time by walking and

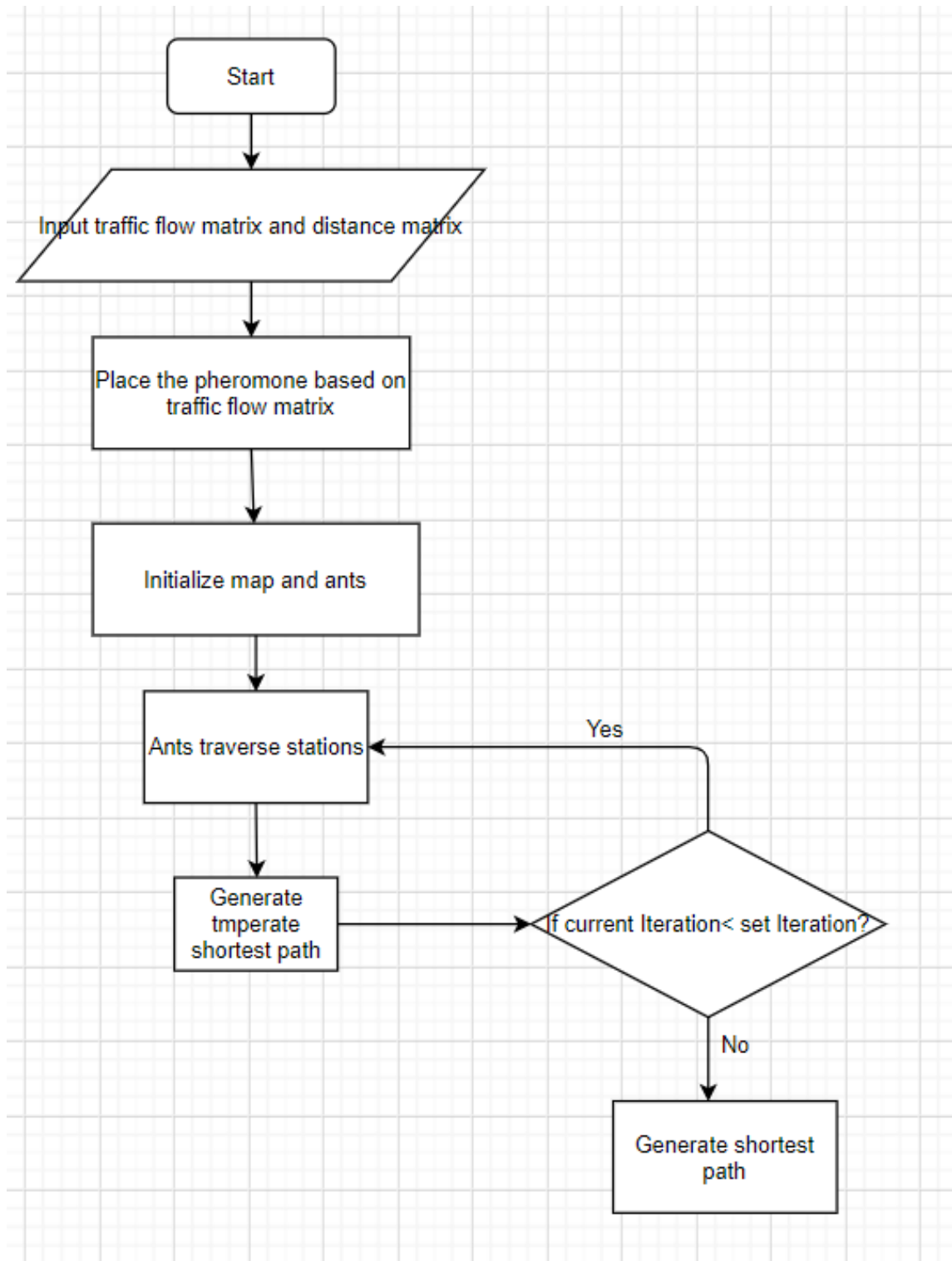


Fig. 5.4: The Flowchart about Traffic Flow Enhancement Algorithm

store it. Subsequently, we assume taking trams is beneficial. Every passenger tries to take the tram as much as possible. When they run the min route and meet the tram-station, they prefer to take tram to the desination rather than walking to the desination. And the passenger will think whether they needs to transfer another bus-line to transfer the bus. In additional, the tram-speed and walk-speed are different. The cumulative time is added to the distance which divides the speeds corresponding different travel way. The final step is comparing the inclusive time between walking and tram. The we can get the different total time at different cases. After that we just select the minimize total time in all possiple cases and outputs the method of cut-lines.

The specific processes can be displayed in the following:

1. Firstly, we cut the one shortest route in all possible cases depended on the requirement of routes needed.
2. Secondly, every passenger, depend on A* algorithm, we calculate the time by only walk and the time by hybird.
3. Thirdly, comparing the two times and select the smaller one as the final criteria for the passenger.
4. Subsequently, calculating all the final criteria and adding them as the criterion for this kind of cutting method.
5. After that, calculating criterriion for all cases of cutting methods and compare them.
6. Finally, we select the cutting way with minimum criteria.

The flowchart can be represent like this Fig.5.5.:

5.6 OpenGL Visualization

The images are visualized through OpenGL-2D. During the process, the C++ program read the data from csv file(Fig.5.6.), which created by enhanced-ant-colon-algorithm by python. Subsequently, the program creates some two-dimensional array to store the data and depends some drawing function to display the entire algorithm demonstration process . After that, we can adjust different parameters to get different visualization results.

The example of the program is 30 tram stations and 20 passengers. The above csv files are developed by the algorithm. These file seperately store the information about traffic flow matrix (another name: bus busy matrix), the line in each iteration by ant-colony algorithm, the stations of cartesian coordinates, and all cut cases for the one shortest line (Fig.5.7.). You can get detailed information in appendix(GitHub).

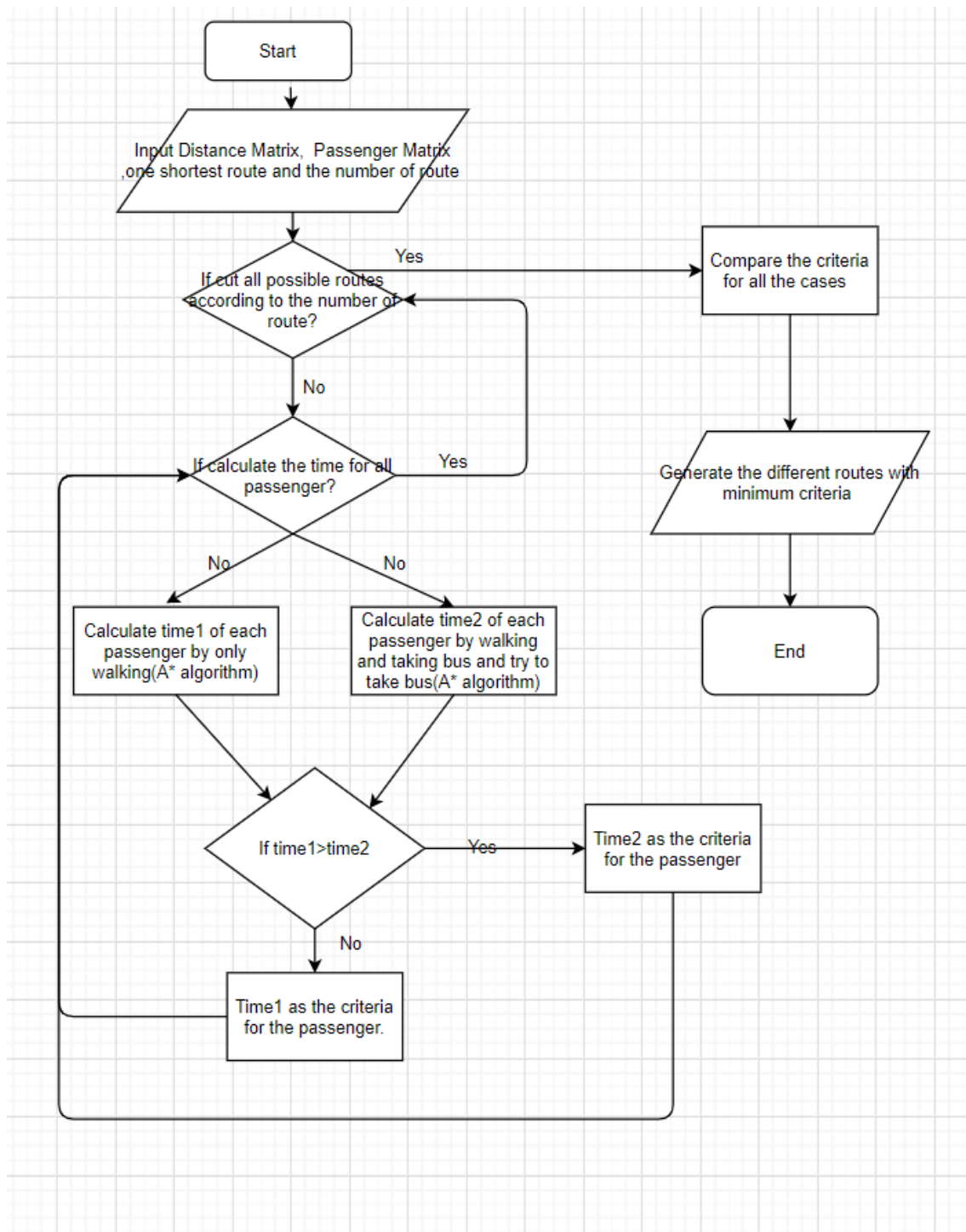


Fig. 5.5: The Flowchart about Cut-lines Algorithm





 bus_busy_matrix.csv	2020/6/4 21:34	Microsoft Excel Co...	4 KB
 lines.csv	2020/6/4 21:34	Microsoft Excel Co...	2 KB
 points_matrix.csv	2020/6/4 21:34	Microsoft Excel Co...	5 KB
 sub_lines.csv	2020/6/4 21:34	Microsoft Excel Co...	1 KB

Fig. 5.6: The Flowchart about Cut-lines Algorithm

(a) traffic flow matrix

(b) One shortest line

(c) stations of cartesian coordinates

(d) cut cases for the one shortest line

Fig. 5.7: csv files

5.7 Data analysis

The R analysis program use the GGplot2 library to analyse the large data sets (note: this data set is a virtual data created by the python algorithm). The purpose for the process is get the relative optimal parameters in enhanced ant colony algorithm rather than other parameter in specific requirements. For example, the Appedix (GITHUB) is the dataset about 20 stations and 30 stations.

dataSet_part1.csv - Excel

文件开始插入页面布局公式数据审阅视图加载项帮助操作说明搜索

X25

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Bus_Numt	Passenger	Iteration	Pheromon	Alpha	Beta	Volatile_C	Ant_Phero	Ants_Num	Shortest_L	ShortestTii	Cut_Lines	Cut_Times							
2	20	30	100	15	3	3	0.4	115	25	[4, 1, 7, 9,	353.0564	[17, 9, 14,	716.8647							
3	20	30	100	15	3	3	0.4	115	30	[2, 6, 9, 3,	362.3253	[4, 1, 13,	714.871							
4	20	30	100	15	3	3	0.4	115	35	[14, 19, 8,	356.5941	[16, 2, 17,	864.9205							
5	20	30	100	15	3	3	0.4	130	25	[7, 9, 4, 1,	332.3839	[11, 14, 19,	779.7504							
6	20	30	100	15	3	3	0.4	130	30	[18, 13, 6,	331.2602	[14, 19, 8,	805.1396							
7	20	30	100	15	3	3	0.4	130	35	[9, 7, 3, 11,	341.8566	[7, 3, 11,	777.9578							
8	20	30	100	15	3	3	0.4	145	25	[9, 12, 0, 1,	346.2692	[11, 5, 3,	760.5322							
9	20	30	100	15	3	3	0.4	145	30	[19, 14, 16,	358.8385	[10, 17, 4,	718.0299							
10	20	30	100	15	3	3	0.4	145	35	[16, 19, 14,	352.7048	[16, 19, 1,	853.6621							
11	20	30	100	15	3	3	0.4	160	25	[8, 14, 19,	360.7883	[14, 19, 1,	801.1031							
12	20	30	100	15	3	3	0.4	160	30	[10, 17, 5,	347.3542	[0, 12, 3,	825.4049							
13	20	30	100	15	3	3	0.4	160	35	[11, 0, 12,	357.5678	[0, 12, 15,	729.8892							
14	20	30	100	15	3	3	0.4	175	25	[18, 15, 14,	335.2112	[19, 8, 3,	767.8017							
15	20	30	100	15	3	3	0.4	175	30	[2, 9, 0, 11,	364.0092	[0, 11, 12,	768.6259							
16	20	30	100	15	3	3	0.4	175	35	[16, 18, 13,	343.8192	[16, 18, 1,	708.3978							
17	20	30	100	15	3	3	0.5	115	25	[0, 12, 11,	351.6962	[11, 5, 10,	795.0106							
18	20	30	100	15	3	3	0.5	115	30	[2, 19, 14,	347.2416	[19, 14, 1,	895.0623							
19	20	30	100	15	3	3	0.5	115	35	[2, 19, 14,	342.9742	[2, 19, 14,	885.9597							
20	20	30	100	15	3	3	0.5	130	25	[1, 4, 6, 13,	335.9327	[18, 9, 7,	793.8169							
21	20	30	100	15	3	3	0.5	130	30	[9, 3, 5, 11,	330.7614	[9, 3, 5, 1,	776.0987							
22	20	30	100	15	3	3	0.5	130	35	[18, 13, 15,	339.1494	[18, 13, 1,	860.4785							
23	20	30	100	15	3	3	0.5	145	25	[18, 13, 6,	336.9132	[6, 1, 4, 7,	812.1771							
24	20	30	100	15	3	3	0.5	145	30	[8, 17, 10,	348.0647	[10, 5, 11,	905.2655							
25	20	30	100	15	3	3	0.5	145	35	[14, 19, 8,	343.5571	[15, 18, 1,	750.6276							
26	20	30	100	15	3	3	0.5	160	25	[8, 19, 14,	338.5991	[14, 2, 16,	863.1246							
27	20	30	100	15	3	3	0.5	160	30	[19, 14, 8,	339.4325	[14, 8, 18,	849.2927							
28	20	30	100	15	3	3	0.5	160	35	[2, 18, 12,	349.1649	[0, 5, 3, 4,	759.2284							
29	20	30	100	15	3	3	0.5	175	25	[3, 0, 12, 1,	339.7368	[12, 11, 7,	797.1055							
30	20	30	100	15	3	3	0.5	175	30	[13, 15, 0,	342.5501	[11, 5, 12,	685.5563							
31	20	30	100	15	3	3	0.5	175	35	[6, 13, 0, 1,	355.1234	[6, 13, 0,	747.6679							
32	20	30	100	15	3	3	0.6	115	25	[7, 11, 0, 1,	322.5614	[0, 12, 10,	779.4282							
33	20	30	100	15	3	3	0.6	115	30	[17, 10, 5,	351.2621	[0, 11, 12,	844.8811							
34	20	30	100	15	3	3	0.6	115	35	[11, 0, 12,	328.986	[8, 19, 14,	714.0405							
35	20	30	100	15	3	3	0.6	130	25	[12, 0, 11,	335.9468	[12, 0, 11,	755.142							
36	20	30	100	15	3	3	0.6	130	30	[6, 13, 14,	325.2917	[6, 13, 14,	791.6913							
37	20	30	100	15	3	3	0.6	130	35	[11, 0, 12,	320.7415	[12, 18, 1,	786.2277							
38	20	30	100	15	3	3	0.6	145	25	[4, 1, 9, 7,	331.345	[3, 12, 0,	805.3568							
39	20	30	100	15	3	3	0.6	145	30	[2, 19, 14,	344.5803	[2, 19, 14,	807.9358							
40	20	30	100	15	3	3	0.6	145	35	[14, 19, 8,	348.8163	[19, 8, 15,	731.682							
41	20	30	100	15	3	3	0.6	160	25	[11, 0, 12,	345.1259	[0, 12, 3,	803.7779							
42	20	30	100	15	3	3	0.6	160	30	[11, 0, 12,	350.7944	[9, 1, 4, 7,	792.2636							
43	20	30	100	15	3	3	0.6	160	35	[9, 7, 4, 1,	324.7293	[7, 4, 1, 0,	792.7673							
44	20	30	100	15	3	3	0.6	175	25	[0, 1, 4, 7,	339.2148	[1, 4, 7, 9,	711.8315							
45	20	30	100	15	3	3	0.6	175	30	[17, 10, 5,	363.5263	[17, 10, 5,	824.802							
46	20	30	100	15	3	3	0.6	175	35	[19, 16, 14,	328.292	[15, 13, 6,	792.6277							
47	20	30	100	15	3	3	0.7	115	25	[5, 10, 17,	347.4931	[8, 15, 6,	831.8467							

Fig. 5.8: The Flowchart about Cut-lines Algorithm

Chapter 6

Implementation

The section describes the implementation of each algorithm and programming technology in detail, including the part of python, OpenGL and R.

6.1 Algorithm Implementation

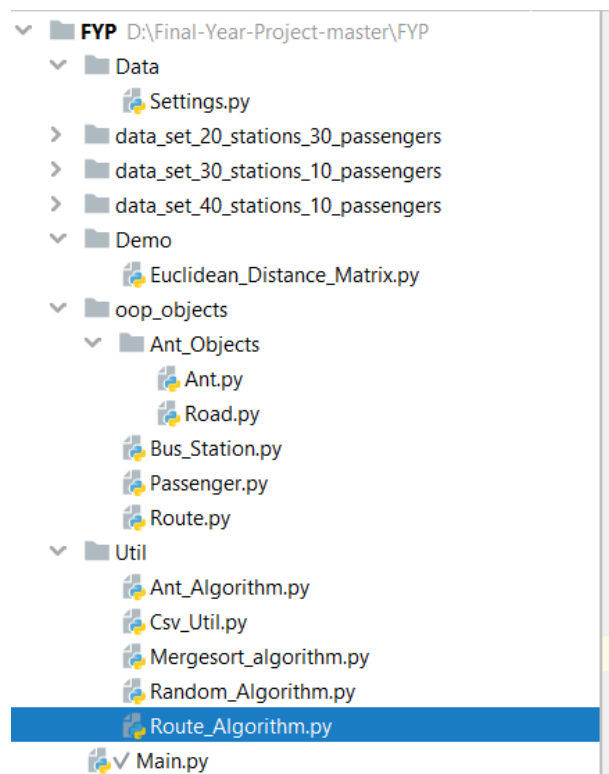


Fig. 6.1: The Flowchart about Cut-lines Algorithm

In order to implement the enhanced ant colony algorithm, the framework of python program has been achieved (Fig.6.1.). Our implementation is divided into two parts. The first part is about parameter adjustment class named Settings, which make the program has very great encapsulation. We only need to change this part of data to modify the

entire data of program and needn't change other parts. The second part is the specific implementation of the algorithm. It has been divided into three small blocks. The first piece is the object oriented part named oop objects. We create all the objects relevant to the correlation algorithm, such as the station object, route object, ant object and path object travelled by the ant. The second chunk named Demo is a model that converts the distance matrix to coordinates by eigenvalue decomposition. The third block named Util is the util classes that use the object to implement various functions, including random data formation(i.e. virtual stations, virtual passengers, virtual ants, etc.), and write the csv files that relevant iteration process or generate a large data set for later parameter analysis. Also, the main part is enhanced ant colony algorithm (traffic flow algorithm, ant colony algorithm and cut-lines Algorithm).

6.1.1 Parameter Adjustment Class

The purpose of class of setting in Data directory is storing a lot parameters about the enhanced ant colony algorithm and random data formation algorithm. In addition, we can easily change different parameters to handle distinct problems according to the requirements without modifying other classes. The benefits about the structure are the robustness of the program and easily to modify. For example, it's convenient for tram company to add another line depended on the original tram maps.

The major architecture of the class is the instance attributes of the parameters:

- seed: the seed of random for the program
- bus-station-number: the cumulative number of the program
- passenger-number: the total number of passenger
- edge-distance: the maximum distance for each stations
- station-matrix: store the matrix of stations
- passenger-matrix: store the matrix of passenger
- bus-busy-matrix(traffic flow matrix): store the information of "food" in the graph
- pheromone-matrix: store the information of pheromone in the graph
- pheromone: each ant contains the pheromone
- alpha: the parameter for the ant-colony-algorithm
- beta: the parameter for the ant-colony-algorithm
- volatile-coefficient: volatilization rate
- ant-pheromone-quality: the inclusive pheromone amount all ants

- Q: same as ant-pheromone-quality
- ants-number: the number of ants
- max-iteration: the maximum of iteration
- bus-speed: the speed of bus
- passengers-speed: the speed of passengers
- cutting-edge: how to cut the edge in different parts

The methods for the class:

- **value—change**(self,pheromone,alpha,beta,volatile-coefficient,ant-pheromone-quality,ant-number): change the parameters.

6.1.2 Random data Input

The random data generation is constructed by these classes includes ***RandomAlgorithm*** class, ***Route***. In summury, we use the random library to assign the random value to the objects and generate the relevent matrixs randomly. The first part is the implementation these functions to the objects, we need to use the Random Algorithm.py to handle these objects, Such as the tram station, passenger, ant.

```

1 import random
2 from Data.Settings import settings
3 from oop_objects.Bus_Station import bus_station
4 from oop_objects.Passenger import passenger
5 from oop_objects.Ant_Objects.Ant import ant
6 from Demo.Euclidean_Distance_Matrix import euclidean_distance_matrix

```

The above code means implmenting the object in the ***RandomAlgorithm***. The function of ***RandomAlgorithm*** is assignning the random value to the objects which need the the random value. Later on, if there are the real value in the program and it's fine to assign them to the objects. The instance of class can be observed as:

- setting: the configured attribution for the system.
- ants-name: the name of ant
- new-euclidean:the object that generate the distance matrix

And the methods for the class:

- random-passenger(number): generte the number of passengers for tram stations
- random-station(number): generate the name for the random stations which is helpful for th distance matrix

- random-edge(stations): generate the different random edge for the stations
- random-ants(ants-number,bus-station,settings):generate random ants that walk around in the graph, which is useful for the ant-colony algorithm

The next part is the class of route. One part of function by the route is how to input the data to the martrix. As the following code, the objects always stored as list or direction at first. And by the we create a matrix to store their information initiatively. Such as the station matrix and passenger matrix, they both transfered by the information of objects.

The instance of class can be observed as:

- bus-stations: the list of bus-stations to store the objects of buses
- passengers: the list of passengers to store the objects of passenger
- settings: the configured attribution for the system.

And the methods for the class:

- start-route(): the random passengers and stations were stored in the *bus – stations* and *passengers*
- bus-stations-matrix(): convert the information from *bus – stations* to *bus – stations – matrix*
- passengers-matrix(): convert the information from *passengers* to *passengers – matrix*

After finishing all the above classes, we could get the input stream for the program. The Distance matrix can be observed like Fig.6.2.The Fig.6.3 represent the passenger matrix.

```
The distance between different bus station :
[[ 0.          62.80127387 62.16912417 49.92995093 62.96824597]
 [62.80127387  0.          18.02775638 42.05948169 44.91102315]
 [62.16912417 18.02775638  0.          55.78530272 61.07372594]
 [49.92995093 42.05948169 55.78530272  0.          13.03840481]
 [62.96824597 44.91102315 61.07372594 13.03840481  0.          ]]
```

Fig. 6.2: Distance matrix

```
The location information for passengers :
[[0. 0. 0. 0. 0.]
 [1. 0. 1. 0. 0.]
 [0. 2. 0. 0. 0.]
 [0. 0. 1. 0. 2.]
 [2. 0. 1. 0. 0.]]
```

Fig. 6.3: Passenger matrix

6.1.3 Enhanced ant colony algorithm

The Enhanced ant-colony algorithm is the core part of this project. The algorithm can be divided into three parts, which are traffic flow enhancement, ant-colony algorithm and cut-lines algorithm. Firstly, the implementation of traffic flow enhancement will be achieved. The algorithm is constructed by three main class, including the bus-station class, part of route class and mergesort-algorithm. Now we analyze the implementation step by step.

As for the class of bus-station, it represents one station in the graph. The instance of class can be satisfied with the formula(4.6), each station as one node of graph has the value of $G(\text{Station})$ and $H(\text{Station})$..:

- g: for the location of station, the length of path from start location to current location.
- h: for the location of station, the expected path from current location to the destination.
- f: the value combined g and h.

And the methods for the class:

- get-g(value): assign the value of g to the station
- add-g(value): add the value g
- get-h(value): assign the value of h to the station
- result-f(): f equal to f plus g
- clean-data(): delete all the value of the station

```
1 class bus_station():
2     '''the class represents the bus station'''
3     def __init__(self):
4         #A* algorithm
5         self.g = 0
6         self.h = 0
7         self.f = 0
8
9     def get_g(self, value):
10        '''get the value of g'''
11        self.g = value
12
13    def add_g(self, value):
14        '''add the value of g'''
15        self.g = self.g + value
16
17    def get_h(self, value):
18        '''get the value of h'''
19        self.h = value
20
21    def result_f(self):
```

```

22         '''print the value of f '''
23         self.f = self.g+self.h
24
25     def clean_data(self):
26         '''clear the g,h,f to zero'''
27         self.h = 0
28         self.g = 0
29         self.result_f()

```

The stations with value of H and G have been created from the above code.

As for the part of route class, it generates the matrix named traffic flow matrix(bus-busy-matrix). Now we look at its function method:

- assign-value-bus-busy-matrix: assign the station in shortest line to the traffic flow matrix
- busy-route-matrix: generate the final traffic flow matrix

```

1     def busy_route_matrix(self):
2         '''generate the bus_busy_route matrix'''
3
4         def assign_value_bus_busy_matrix(self ,bus_busy_matrix ,magnitude ,tmp_route)
5             :
6             '''assign the value to the traffic flow matrix'''

```

We can use the codes to implement the A^* algorithm for each passenger who has requirements to the destination.

The class is about the part of route algorithm. The function is to execute the a^* algorithm specifically. The instance attributes about the class:

- settings: the configured attribution for the system.
- open-list: act as a container to store the station you want to extend next
- flag-find: whether the final destination has been found
- tmp-nodes: store the other nodes needed to traverse
- mergesort-algorithm: the algorithm about merge-sort to rank the value of $F()$ in other nodes.

The instance functions about the class:

- a-star-algorithm(start-location, dest-location, bus-stations): search the best route for each passenger to the destination
- valuation-station(successor, dest-location, bus-stations): initialize all the $f(g+h)$ to the specific nodes
- a-brain-judge-1(dest-location, bus-stations): whether the direct-line is the optimize

- sort(bus-stations): use merge-sort to sort the other station based on value of $F()$
- clean-f(stations): clean the all the value about g, h and f in stations.

```

1 from Util.Mergesort_algorithm import mergesort_algorithm
2 import numpy as np
3 class route_algorithm():
4     '''generate the route to optimise the profiles'''
5
6     def __init__(self, settings):
7         self.settings = settings
8
9         # a* algorithm sequence
10        self.open_list = []
11        self.route = []
12
13        #a* brain_judge
14
15        self.flag_find = False
16        self.tmp_nodes = []
17
18        #print_parameter
19        self.lines = []
20
21        # merge_sort algorithm
22        self.mergesort_algorithm = mergesort_algorithm()
23
24
25        def A_STAR_Algorithm(self, start_location, dest_location, bus_stations):
26            '''search the best route for each passenger to the destination'''
27        ....
28        def voluation_station(self, successor, dest_location, bus_stations):
29            '''initilize all the f(g+h) to the specific nodes'''
30        ....
31        def a_brain_judge_1(self, dest_location, bus_stations):
32            '''whether the direct_line is the optimize'''
33        ....
34        def sort(self, bus_stations):
35            '''sort all the f in the next stations'''
36        ....
37        def clean_f(self, stations):
38            #clean the all the value about g, h and f.
39        ....

```

The process for the algorithm to can look like this Fig.6.4. The true of flag means :The smallest two values do not conflict and do not need to run Mergesort. And each shortest passenger path is the sequence of the station order. The output of traffic-flow-matrix can be observed as Fig.6.5

Secondly, we implement the original-ant-colony algorithm. The original ant-colony-algorithm is constructed by three parts. There are ant, road and ant algorithm. The following class describe the behavior of ants. There some essential methods for the class:

- initial-city-method(self): search the best route for each passenger to the destination
- change-current-city(self,station-name): change the current city to another city

```

NEXT PASSENGER!-----
True
2
3
NEXT PASSENGER!-----
True
3
0
NEXT PASSENGER!-----
True
3
2

```

Fig. 6.4: Process of Algorithm

```

The number of passengers in the bus_busy_matrix
[[0. 5. 1. 2. 5.]
 [5. 0. 4. 2. 1.]
 [1. 4. 0. 2. 4.]
 [2. 2. 2. 0. 4.]
 [5. 1. 4. 4. 0.]]

```

Fig. 6.5: traffic-flow-matrix

- back-to-initial-city(self): back to the initial city
- select-city(self): choose another city randomly
- calculate-denominator(self): calculate the value of the current denominator in the city.
- calculate-route(self): calculate the inclusive length of the ant.
- calculate-criteria(self): calculate the critira time each ant.

Also there are some core instance attributes for the class:

- passed-city: as the tabu-table, the ant caanot pass the city that already travelled
- name:Ants name
- time: the time it takes to travel all the paths
- route:The route that ants travel

```

1 class ant():
2
3     def __init__(self ,bus_stations ,settings ,ants_name):
4         self.passed_city = []
5
6         #Travel times
7         self.name = ants_name
8
9         #Ant's name
10        self.current_city= 0

```

```

11
12     # in the city
13     self.inital_city = 0
14
15     # Location of the original city
16
17     self.other_cities = []
18
19     self.settings = settings
20
21
22     self.random_rate = 0.0
23
24
25     self.denominator = 0
26
27     #The sum of the values of the other points
28
29     self.route=[]
30     #The route that ants travel
31
32     self.length = 0
33
34
35     self.bus_stations = bus_stations
36
37
38     self.molecular = {}
39     #The value of the ant molecule
40
41
42     self.time = 0
43     #The time it takes to travel all the paths
44
45     def initial_city_method(self):
46         #Initialize ant city
47         .....
48
49         def change_current_city(self ,station_name):
50             #Change the existing city to the present city
51             .....
52
53         def back_to_initial_city(self):
54             .....
55
56         def select_city(self):
57             .....
58
59         def calculate_denominator(self):
60             #Calculate the value of the denominator
61             .....
62
63         def calculate_route(self):
64             #Calculate the total length of the ant
65             .....
66
67     def calculate_criteria(self):
68         #Calculate the time it takes to transport passengers after each ant
69         #crawls its chosen length

```

The class of road construct some basic attributes in the edge and the method of *create – road()* is place the pheromones in the road. You can see it in the following code:

```

1 class road():
2
3     def __init__(self, start_station, end_station, settings):
4         self.start_station = start_station
5         self.end_station = end_station
6         self.distance = 0
7         self.pheromone = 0
8         self.settings = settings
9         self.ant_brain_change_value = 0
10
11
12     def create_road(self):
13         #Generation of corresponding coordinate pheromones

```

The class of ant-algorithm is the main frame for the algorithm, which is considered from the time of iteration in the heuristic approach. We analyze some core instance attributes for the class:

- min-route: as a list to store one shortest line
- shortest-length: store the minimum length during the iteration
- time: store the minimum time during the iteration
- total-passengers-time: after the cut-line algorithm implementing the ant-colony algorithm, the cumulative spend time for passengers

Here are some essential instance function for the algorithm:

- generate-ants: generate a number of ants randomly assigned to the city
- travel-the-cities(self, bus-station): determines whether the iteration ends
- select-method(self, ant): the probability of selecting another city
- update-information-pheromone(self): update the pheromone in the matrix
- add-foods(self): replace the pheromone from enhanced traffic flow algorithm
- calculate-mini-route(self): get the shortest route

```

1 import numpy as np
2
3 from Util.Random_Algorithm import random_algorithm
4 class ant_algorithm():
5
6     def __init__(self, settings):
7
8         #The shortest path
9         self.shortest_lines = []

```



```

10     self.shortest_length = 999999
11     self.time = 999999
12
13     self.random_algorithm_ant = random_algorithm()
14
15
16     self.ants = []
17
18
19     #The shortest path required to traverse all cities
20     self.min_route = []
21
22
23     self.settings = settings
24     '''Settings'''
25
26     self.Q = settings.Q
27
28     self.ants_number = settings.ants_number
29
30     self.max_iteration = settings.max_iteration
31
32
33     self.sub_line = []
34
35
36     self.total_passengers_time = 0
37
38
39     self.shortest_cut_lines = []
40
41
42     self.cut_time = 999999
43
44
45     self.csv_lines = []
46     self.csv_line = []
47
48
49     self.csv_sub_lines = []
50     self.csv_sub_line = []
51
52
53     def generate_ants(self, bus_station):
54         #Generate a number of ants randomly assigned to the city
55         ....
56
57     def travel_the_cities(self, bus_station):
58         '''Determine if the number of iterations is used up'''
59         .....
60
61     def select_method(self, ant):
62         #Select the next location that you randomly go to
63         ....
64
65     def update_information_pheromone(self):
66         #Update the road pheromone
67         ....
68
69     def add_foods(self):

```

```

70     #Add food to make ants produce more pheromones in the path of food , build a
        pheromone matrix
71     ....
72
73     def calculate_mini_route(self):
74         '''calculate the tmp_shortest route'''
75     .....
```

The final part is talking about the cut-line algorithm. It locates in the ant-algorithm. The core algorithm methods are:

- cutting-edge(self,line-number,passenger-route): cut the route in all possible.
- judge-whether-lines(self,start-location,dest-location,route): whether the tram can direct arrive the dest from the start
- calculate-mini-cost-passenger(self,passenger-route): calculate the total time for all the passengers.

The process of ant-colony algorithm in the programm like this Fig.6.6. Each ants serach the stations randomly and compare the spend time with the minimize route. And if the time of ant's route is smaller, then taking the place of the minimize time.

```

the name of ant: 8 :-->3-->0-->1-->2-->4
the name of ant: 9 :-->2-->1-->0-->4-->3
the name of ant: 10 :-->0-->1-->2-->3-->4
the name of ant: 11 :-->1-->2-->3-->4-->0
the name of ant: 12 :-->2-->0-->1-->4-->3
the name of ant: 13 :-->4-->0-->1-->2-->3
the name of ant: 14 :-->1-->0-->4-->3-->2
the name of ant: 15 :-->0-->4-->3-->2-->1
the name of ant: 16 :-->1-->2-->3-->4-->0
the name of ant: 17 :-->1-->2-->3-->4-->0
the name of ant: 18 :-->1-->2-->3-->4-->0
the name of ant: 19 :-->2-->1-->0-->4-->3
the name of ant: 20 :-->3-->4-->0-->1-->2
149.63064869676015
The tmp_shortest route is: -->4-->3-->0-->1-->2-->4
```

Fig. 6.6: ant-colony

```

1     def cutting_edge(self ,line_number , passenger_route):
2         '''cutting the edge to the different route'''
3     ....
4
5     def judge_whether_lines(self , start_location , dest_location , route):
6         , ,
7         #Compute two points in the same path
8
9     def calculate_mini_cost_passenger(self , passenger_route):
10    #According to the requirements of the transfer , three schemes are used to
        work out which one takes the shortest
        time to reach the required point
11    .....
```

All the specific codes can be found in the appendix(GitHub).

The result of cut-line algorithm in the programm like this Fig.6.7. Simulating each passenger take the bus to the destination. And if he needs to transfer another route of tram, it prints the "the passenger need to change the bus!". The final lines like the end.

```

the passenger need to change the bus!
the passenger need to change the bus!
the passenger need to change the bus!
the passenger need to change the bus!
the passenger need to change the bus!
the passenger need to change the bus!
the passenger need to change the bus!
490.3294767326911
The sub line is 4->
The sub line is 3->
The sub line is 0->
The sub line is 1->
The sub line is 2->

```

Fig. 6.7: cut-line-algorithm

6.2 Visualization Implementation

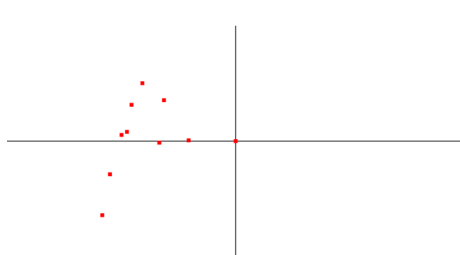
For the realization of data visualization, we have primarily completed two parts that reading data from csv file and using OpenGL to show the production process of enhanced ant colony algorithm. The process can represent like this:

1. From the beginning, the tram stations means generating the coordinate points.
2. Subsequently, the traffic flow algorithm has been finished which means the thicker and darker the line, the more frequent for the passengers travel in. We can add "food" symbolically that means adding amount of pheromones based on the frequency in advance. The higher for the frequency, the more pheromones are there.
3. Furthermore, we implement the original ant colony algorithm whose metric is time to generate one optimal route.
4. Moreover, the cut-lines Algorithm has been implemented. For distinct cutting methods, the transportations for the same passenger are different. Some passengers need to take the tram, some passengers arrive the destination by walking. However, most of passengers transfer the trams (mixed walking and trams).
5. We could figure out the optimal cutting mode for the route to satisfy the cheapest allocation for all the passengers.

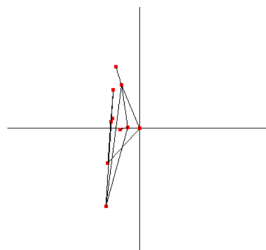
As for the visualization of 10 points and 20 passengers, we can observed the pictures(Fig.6.8.).

As for the visualization of 15 points and 30 passengers, we can observed the pictures(Fig.6.9.).

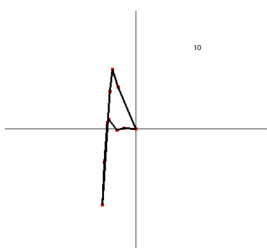
As for the visualization of 40 points and 40 passengers, we can observed the pictures(Fig.6.10.).



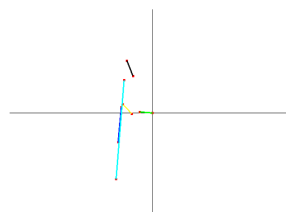
(a) stations of cartesian coordinates



(b) traffic flow matrix

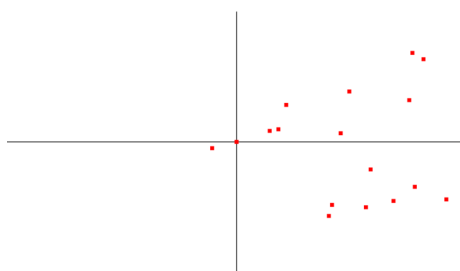


(c) One shortest line

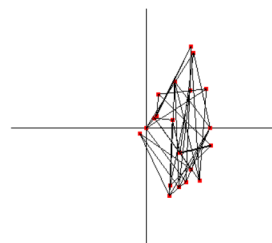


(d) cut cases for the one shortest line

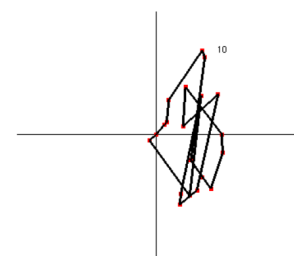
Fig. 6.8: 10 points and 20 passengers



(a) stations of cartesian coordinates



(b) traffic flow matrix



(c) One shortest line



(d) cut cases for the one shortest line

Fig. 6.9: 15 points and 30 passengers

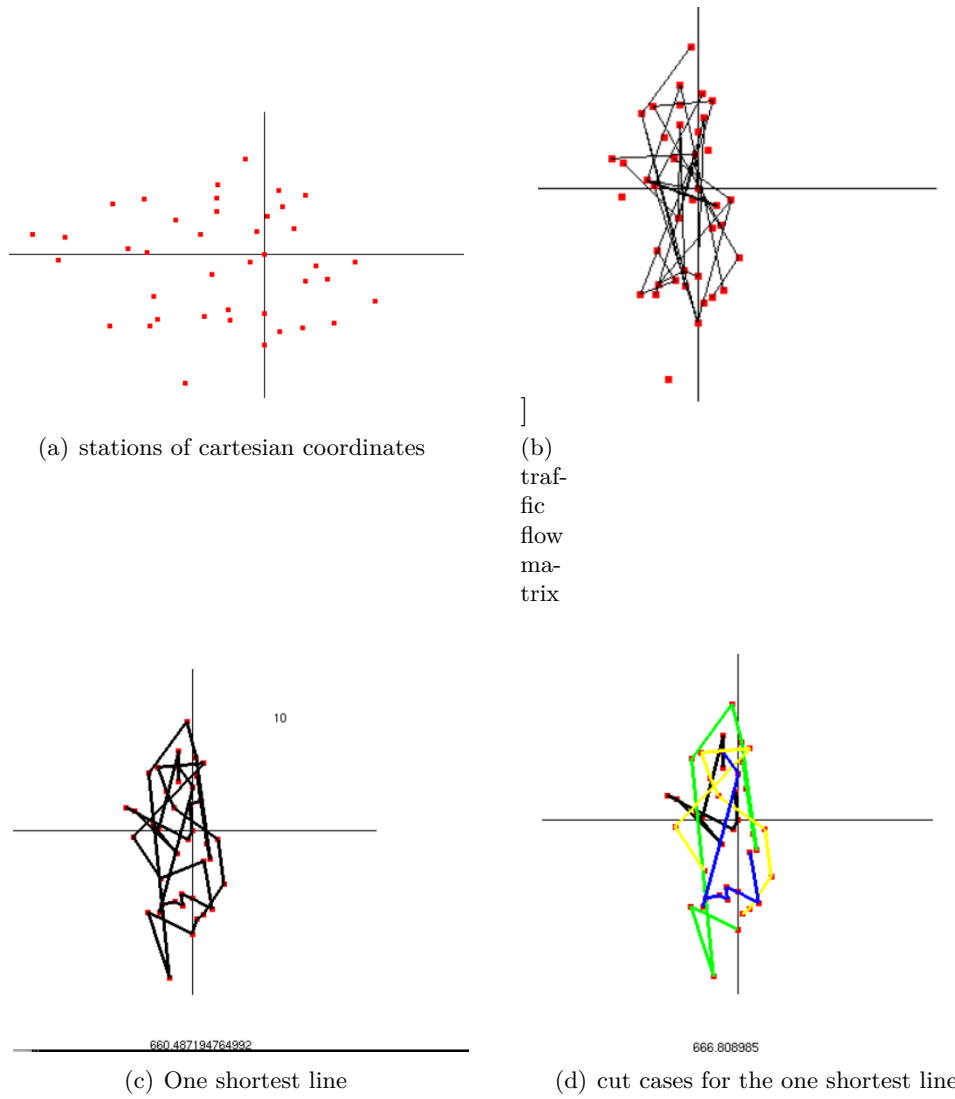


Fig. 6.10: 40 points and 40 passengers

As for the visualization of 50 points and 100 passengers, we can observed the pictures(Fig.6.11.). The OpenGL code will be in the appendix(GitHub).

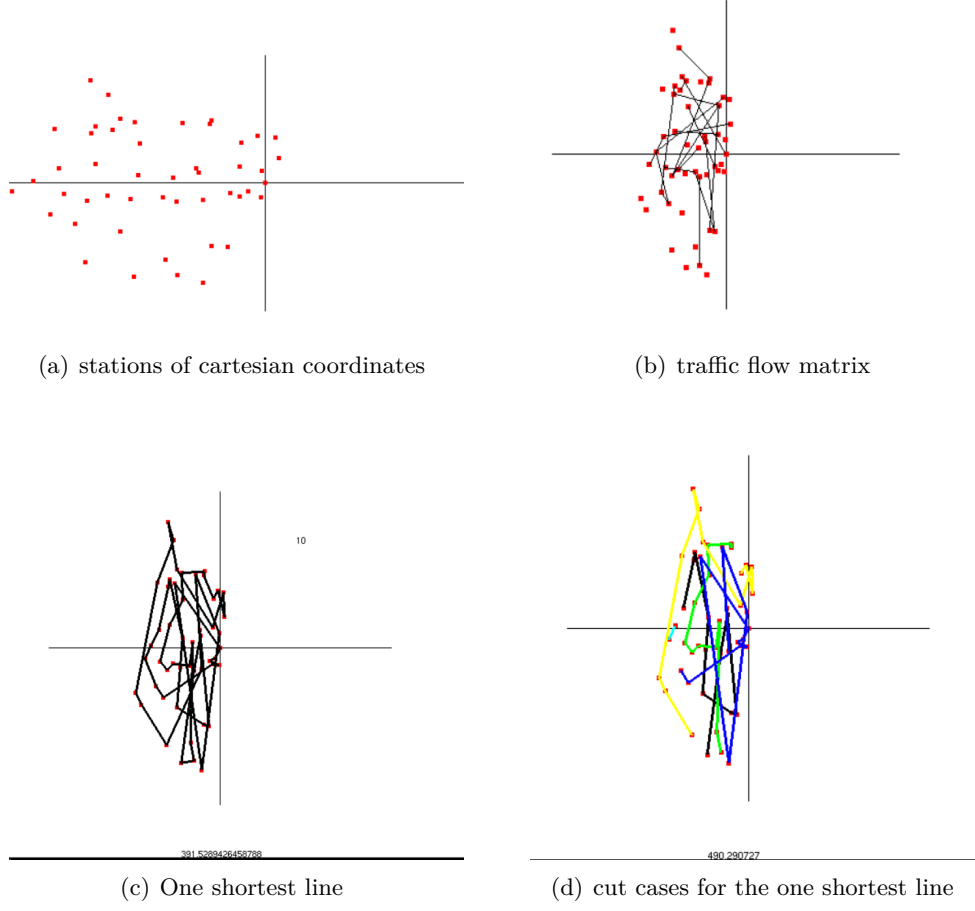


Fig. 6.11: 50 points and 100 passengers

Chapter 7

Evaluation

7.1 Test Background

The section will talk about how to verify the enhance algorithm is more efficient than the traditional ant-colony-algorithm. The traditional ant-colony loss the part of enhanced traffic flow algorithm. Before we compare the two different algorithm. Firstly, we need to adjust the parameters to the optimal parameters in specific case. Then we compare the results for different algorithm and based on the results we conclude the summary that the enhanced ant-colony-algorithm is more efficient than the traditional ant-colony-algorithm.

7.2 Data Analysis

For the data analysis level, we may meet the actual requirements. For example, if the bus company gives you the requirements of 20 stations and 30 passengers, we will need to debug the optimal algorithm parameters for the real situation. At this time, we need to analyze a large number of data sets and finally use a library of R language(GGPlot2) to make visual graphics, and then we can intuitively select the required parameters. The dataset has the same number of each parameters and the code like this Fig.7.1, so we don't worry about the imbalance in the amount of data of parameters. And it also the basics of the following test section.

The following picture corresponding the order of optimal parameters. (Fig.7.2.).

1. Ant-Pheromone-Quality, by observing the picture, 145 is the suitable value.
2. Alpha, by observing the picture, 3 is the suitable value.
3. Ants-Number, by observing the picture, 30 is the suitable value.
4. Beta, by observing the picture, 6 is the suitable value.
5. Pheromone, by observing the picture, 60 is the suitable value.
6. Volatile-Coefficient , by observing the picture, 0.8 is the suitable value.

```

def calculate_twenty_stations_part4():
    '''parameter'''
    pheromone = 10.0

    new_route = route()
    new_route.start_route()
    new_csv_util = csv_util()

    print("The distance between different bus station :")
    new_route.bus_stations_matrix()

    print("The location information for passengers :")
    new_route.passengers_matrix()
    settings = new_route.busy_route_matrix()
    passenger_route = new_route.passengers_route

    # Write the data to a CSV file
    f = open("dataSet_part1.csv", "w", encoding='utf-8', newline='')
    csv_writer = csv.writer(f)
    csv_writer.writerow(
        ["Bus_Numbers", "Passenger_Number", "Iteration_Times", "Pheromone", "Alpha", "Beta", "Volatile_Coefficient",
        "Ant_Pheromone_Quality", "Ants_Number", "Shortest_Line", "shortest_Time", "Cut_Lines", "Cut_Times"])

    # Set the value of each item manually to let the ant colony algorithm run
    for pheromone_number in range(3):
        pheromone = pheromone + 5
        alpha = 2
        for alpha_number in range(5):
            alpha = alpha + 1
            beta = 2

            for beta_number in range(5):
                beta = beta + 1
                Volatile_coefficient = 0.3
                for coefficient_number in range(5):
                    Volatile_coefficient = Volatile_coefficient + 0.1
                    Q = 100.0
                    ant_pheromone_quality = 100.0
                    for Q_number in range(5):
                        Q = Q + 90
                        ant_pheromone_quality = ant_pheromone_quality + 15
                        ants_number = 20
                        for ant in range(3):
                            ants_number = ants_number + 5
                            settings.value_change(pheromone, alpha, beta, Volatile_coefficient,
                                                    ant_pheromone_quality, ants_number)

                            new_ant_algorithm = ant_algorithm(settings)
                            new_ant_algorithm.travel_the_cities(new_route.bus_stations)

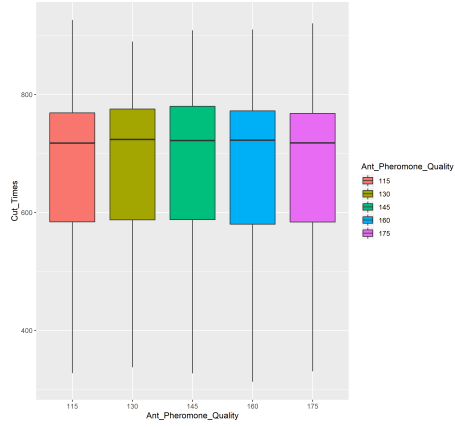
                            new_ant_algorithm.cutting_edge(4, passenger_route)
                            new_ant_algorithm.print_final_cut_lines()

                        csv_writer.writerow([str(settings.bus_station_number), str(settings.passengers_number),
                        str(settings.max_iteration), str(pheromone), str(alpha), str(beta),
                        str(Volatile_coefficient), str(ant_pheromone_quality), str(ants_number)
                        , str(new_ant_algorithm.shortest_lines),
                        str(new_ant_algorithm.time), str(new_ant_algorithm.shortest_cut_lines),
                        str(new_ant_algorithm.cut_time)])

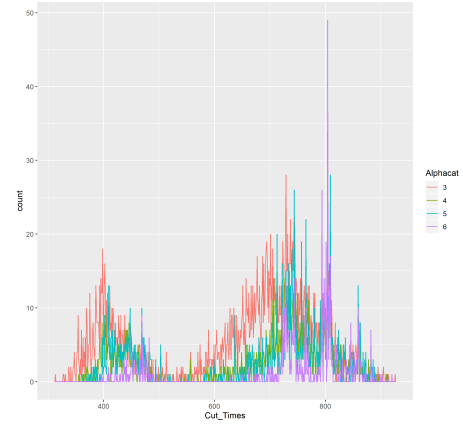
    f.close()

```

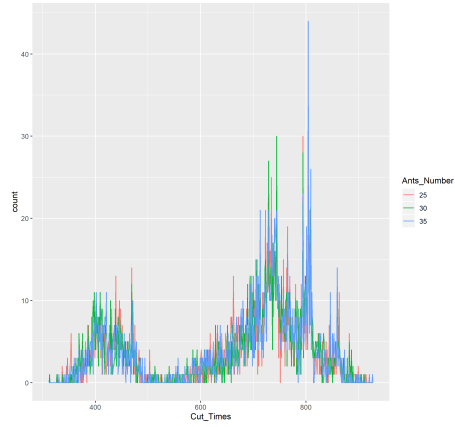
Fig. 7.1: Code generate the Dataset



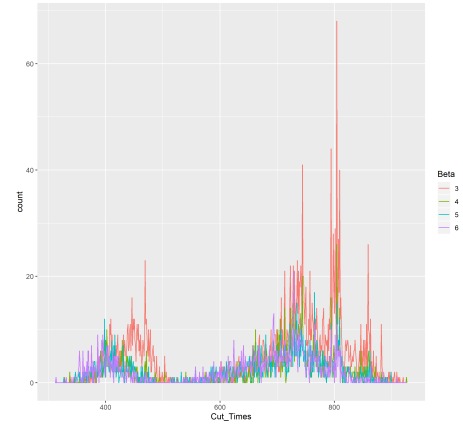
(a) Ant-Pheromone-Quality



(b) Alpha

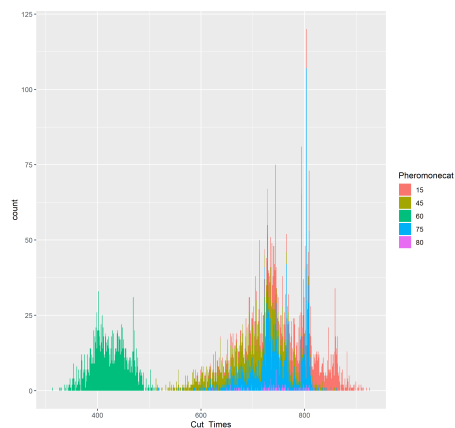


(c) Ants-Number

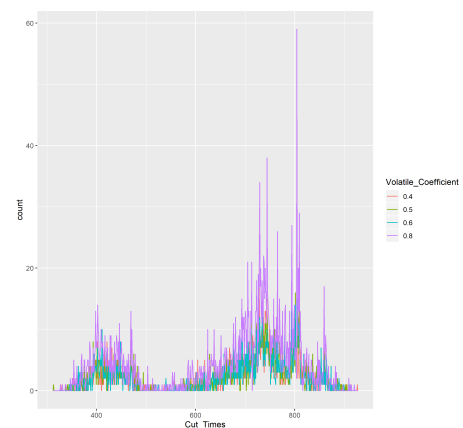


(d) Beta

Fig. 7.2: the order of optimal parameters



(a) Pheromone



(b) Volatile-Coefficient

Fig. 7.3: the order of optimal parameters

7.3 Test Result

In the example, the data of demo is virtual of 20 stations and 30 passengers. Firstly, we generate virtual data to analyze the problem. Both algorithm have same cartesian coordinates of stations and Passengers with similar requirements. Subsequently, we needs to generate the suitable different paramters for each algorithm 4 lines by R. Then, we compare the criteria of two diffenert algorithm in same question with same 100 iterations. The final result will be displayed in the Table.7.4.

	Ant-Pheromone-Quality Alpha	Beta	Ants-Number	Pheromone	Volatile-Coefficient	Max-Iteration	Shortest_Line	shortestTime	Cut_Lines	Cut_Times
Enhanced Ant-Colony Algorithm	145	3	6	30	50	0.8	100 [9, 12, 0, 11, 5, 3, 7, 4, 1, 18, 13, 6, 2, 19, 14, 15, 8, 16, 17, 10, 9]	346.2692227	[[11, 5, 3, 7, 4], [1, 18, 13, 6, 2], [19, 14, 15, 8, 16], [17, 10, 9, 12, 0]]	760.5322
Original Ant-colony Algorithm	140	3	6	30	70	0.6	100 [8, 14, 19, 16, 2, 15, 13, 18, 3, 5, 0, 11, 12, 7, 4, 1, 9, 6, 10, 17, 8]	360.7883234	[[14, 19, 16, 2, 15], [13, 18, 3, 5, 0], [11, 12, 7, 4, 1], [9, 6, 10, 17, 8]]	801.1031

Fig. 7.4: 20 stations and 30 passengers

And we find the cut-time of the enhanced ant-conlony algorithm is smaller then the ant-colony algorithm. Subsequently, we analyse the demo of 30 stations and 10 passengers in the in the Table.7.5 and 40 stations and 10 passengers in the Table.7.6. And for 40

	Ant-Pheromone-Quality Alpha	Beta	Ants-Number	Pheromone	Volatile-Coefficient	Max-Iteration	Shortest_Line	shortestTime	Cut_Lines	Cut_Times
Enhanced Ant-Colony Algorithm	80	5	6	35	50	0.5	100 [12, 0, 18, 15, 1, 8, 24, 3, 14, 10, 27, 4, 19, 7, 5, 26, 16, 11, 29, 22]	131.9288803	[[24, 3, 14, 10, 27, 4, 19], [7, 5, 26, 16, 11, 29, 22], [23, 17, 6, 20, 13, 2]]	171.3349
Original Ant-colony Algorithm	70	4	6	30	80	0.7	100 [21, 6, 9, 2, 15, 1, 24, 8, 12, 25, 0, 18, 17, 19, 13, 7, 5, 3, 14, 10, 4]	140.8868136	[[1, 24, 8, 12, 25, 0, 18], [17, 19, 13, 7, 5, 3, 14], [10, 4, 27, 16, 26, 20], [22, 7, 7, 7, 7]]	223.7707

Fig. 7.5: 30 stations and 10 passengers

stations and 10 passengers in the Table.7.6.

	Ant-Pheromone-Quality Alpha	Beta	Ants-Number	Pheromone	Volatile-Coefficient	Max-Iteration	Shortest_Line	shortestTime	Cut_Lines	Cut_Times
Enhanced Ant-Colony Algorithm	115	8	4	45	45	0.4	100 [36, 0, 3, 14, 20, 4, 6, 18, 28, 22, 13, 12, 26, 32, 15, 31, 35, 25, 10]	320.3234207	[[0, 3, 14, 20, 4, 6, 18, 28, 22, 13], [12, 26, 32, 15, 31, 35, 25, 10, 8, 33]]	427.6981
Original Ant-colony Algorithm	100	9	3	40	50	0.5	100 [23, 0, 3, 14, 1, 11, 24, 37, 2, 36, 39, 16, 21, 29, 30, 35, 4, 12, 18, 1]	313.3528875	[[36, 39, 16, 21, 29, 30, 35, 4, 12, 18], [22, 28, 6, 32, 15, 31, 26, 17, 25, 488.0076]]	488.0076

Fig. 7.6: 40 stations and 10 passengers

By comparing all the demos of cut time, we find that the time of enhanced ant-colony algorithm always smaller Afterwards, ant-colony algorithm. Thus the conclusion is that the enhance algorithm amplifies the original algorithm and become more beneficial for the tram design problem.

Chapter 8

Learning Points

Working on this project has given me a better understanding of how to conduct research and design new approaches to solving problems in the field of computer science and technology. In this part, the useful things that I have learned from this project are introduced. The project about optimization problem need to optimize the tram route in real life. In order to simulate the scene of passenger take the tram routes, I learned how to use the mathematical models to construct practical problems. Furthermore, for settling down the NP problem during construct the model, two heuristic methods were applied in my project whose process make me understand the algorithm deeply. The first approach is the A* algorithm whose purpose is handling the shortest path. However, comparing the traditional dijkstra algorithm, it's more forward-looking and intelligent to seek the shortest path. Then, the ant-colony algorithm is another approach, which simulates the behavior of ants finding the food. It's easily use the meta-heuristic method to solve the np problem like allocating the passengers in the tram stations. Also, there are related mathematical problems to solve, such as how to transform the distance matrix into the actual Cartesian coordinates and how to analyze a set of data sets to obtain the optimal parameter for a problem.

As the time is constrained, I need to manage my project properly so that it can be completed before the deadline. I learned the project manage skills in software development. I used increment model in software development. The project was divided into multiple components. Each component is designed and built separately. The whole project is designed, implemented and tested incrementally. With this learned skill, I can implement every component on time during the development of this project.

Chapter 9

Professional Issues

I confirm that this project obey the Code of Practice and Code of Conduct issued by British Computer Society, and I believe that the whole project is designed to serve the public interest. I guarantee that the whole project is completed by myself. Furthermore, this project is believed to have the ability to benefit other people. In the process of carrying out this project, I have followed the "Key IT practices" stated in the code of practices

Chapter 10

Appendix

The code implementation:

GitHub: ***<https://github.com/MinglangTuo/CSE305>***

The report contains the reference:

- About the tram station of Suzhou[1]
- About application design for the tram company[2]
- About history of combinatorial [3]
- About heuristic method[4]
- About heuristic method[5]
- About heuristic method[6]
- About heuristic method[7]
- About heuristic method[8]
- About heuristic method[9]
- About heuristic method[10]

- About heuristic method[11]
- About P and Np[12]
- About TSP[13]
- About linear algebra[14]
- About A* algorithm[15]
- About ACO[16]
- About OpenGL[17]
- About GGplot2[18]
- About software engineer[19]
- About software engineer[20]

Reference

- [1] C.-L. Chen, “Tram development and urban transport integration in chinese cities: A case study of suzhou,” *Economics of Transportation*, vol. 15, pp. 16 – 31, 2018.
- [2] M. Pérez, R. Loaiza, P. Flores, C. Peralta, and O. Ponce, “A heuristic algorithm for the routing and scheduling problem with time windows: A case study of the automotive industry in mexico.” *Algorithms*, vol. 12, no. 5, 2019.
- [3] A. Schrijver, “On the history of combinatorial optimization (till 1960),” in *Discrete Optimization*, ser. *Handbooks in Operations Research and Management Science*, K. Aardal, G. Nemhauser, and R. Weismantel, Eds. Elsevier, 2005, vol. 12, pp. 1 – 68.
- [4] B. Nguyen, B. Xue, P. Andreae, H. Ishibuchi, and M. Zhang, “Multiple reference points-based decomposition for multiobjective feature selection in classification: Static and dynamic mechanisms.” *IEEE Transactions on Evolutionary Computation*, *Evolutionary Computation*, *IEEE Transactions on*, *IEEE Trans. Evol. Computat*, vol. 24, no. 1, pp. 170 – 184, 2020.
- [5] J. K. Mandal, S. Mukhopadhyay, and P. Dutta, *Multi-Objective Optimization. [electronic resource] : Evolutionary to Hybrid Framework*. Springer Singapore, 2018.
- [6] D. Jones and M. Tamiz, *Practical Goal Programming*. [electronic resource]., ser. *International Series in Operations Research and Management Science*: 141. Springer US, 2010.
- [7] M. A. Jamil, M. K. Nour, A. Alhindi, N. S. Awang Abhubakar, M. Arif, and T. F. Aljabri, “Towards software product lines optimization using evolutionary algorithms.” *Procedia Computer Science*, vol. 163, no. 16th Learning and Technology Conference 2019 Artificial Intelligence and Machine Learning: Embedding the Intelligence, pp. 527 – 537, 2019.
- [8] S.-M. Guua and Y.-K. Wu, “Two-phase approach for solving the fuzzy linear programming problems,” *Fuzzy Sets and Systems*, vol. 107, no. 2, pp. 191 – 195, 1999.
- [9] M. Visue, J. Teghem, M. Pirlot, and E. Ulungu, “Two-phases method and branch and bound procedures to solve the bi-cobjective knapsack problem.” *Journal of Global*

Optimization: An International Journal Dealing with Theoretical and Computational Aspects of Seeking Global Optima and Their Applications in Science, Management, and Engineer, vol. 12, no. 2, p. 139, 1998.

- [10] M. A. Ardakan and M. T. Rezvan, “Multi-objective optimization of reliability”credundancy allocation problem with cold-standby strategy using nsga-ii,” Reliability Engineering and System Safety, vol. 172, pp. 225 – 238, 2018.
- [11] D. Zouache, A. Moussaoui, and F. B. Abdelaziz, “A cooperative swarm intelligence algorithm for multi-objective discrete optimization with application to the knapsack problem,” European Journal of Operational Research, vol. 264, no. 1, pp. 74 – 88, 2018.
- [12] F. Dugardin, F. Yalaoui, and L. Amodeo, “New multi-objective method to solve reentrant hybrid flow shop scheduling problem,” European Journal of Operational Research, vol. 203, no. 1, pp. 22 – 31, 2010.
- [13] A. Dubickas and I. Pritsker, “Extremal problems for polynomials with real roots,” Journal of Approximation Theory, vol. 253, p. 105376, 2020.
- [14] A. J. Maren, “10 - vector-matching networks,” in Handbook of Neural Computing Applications, A. J. Maren, C. T. Harston, and R. M. Pap, Eds. Academic Press, 1990, pp. 141 – 153.
- [15] C. Chen, J. Cai, Z. Wang, F. Chen, and W. Yi, “An improved a* algorithm for searching the minimum dose path in nuclear facilities,” Progress in Nuclear Energy, vol. 126, p. 103394, 2020.
- [16] M. Dorigo and G. Di Caro, “Ant colony optimization: a new meta-heuristic,” in Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), vol. 2, 1999, pp. 1470–1477 Vol. 2.
- [17] “Copyright,” in Advanced Graphics Programming Using OpenGL, ser. The Morgan Kaufmann Series in Computer Graphics, T. McREYNOLDS and D. BLYTHE, Eds. San Francisco: Morgan Kaufmann, 2005, p. iv.
- [18] J. Eckroth, “A course on big data analytics,” Journal of Parallel and Distributed Computing, vol. 118, pp. 166 – 176, 2018.
- [19] R. F. Schmidt, “Chapter 19 - software implementation,” in Software Engineering, R. F. Schmidt, Ed. Boston: Morgan Kaufmann, 2013, pp. 323 – 333.
- [20] G. Bergmann, Á. Horváth, I. Ráth, D. Varró, A. Balogh, Z. Balogh, and A. Ökrös, “Incremental evaluation of model queries over emf models,” in Model Driven Engineering Languages and Systems, D. C. Petriu, N. Rouquette, and Ø. Haugen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 76–90.