VEHICLE ROUTING PROBLEM WITH LOADING CONSTRAINS

By

A1655611 Xuanyu Zhao

*Team Members: A1657343 Yawen Luo A1657881 Wei Zhang*

A THESIS SUBMITTED TO UNIVERSITY OF ADELAIDE

IN PARTIAL FULFILLMENT OF THE DEGREE OF MASTER OF COMPUTING AND INNOVATION

DEPARTMENT OF COMPUTER SCIENCE

JUNE 2015



Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

———————————————————————

Xuanyu Zhao

# Acknowledgements

First and foremost I’d like to thank Dr Sergey Polyakovskiy, Dr Amali Weerasinghe and Mr Kevin MacIunas, for presenting us with the opportunity to take on this project and the guidance they provided.

Following this I would like to thank the people who have formally and / or informally supported us over the MCI project course of the semester: This includes my family, my group members, MCI classmates and my friends.

# Abstract

Our project addresses the heterogeneous fleet vehicle routing problems with two-dimensional loading constraints (2L-HFVRP). The 2L-HFVRP is a combination of the two most prominent problems in distribution logistics, which are the loading of freight into different types of vehicles, and the successive routing of the vehicles to satisfy customers’ demand. The objective of 2L-HFVRP is to achieve the minimum total distance of all routes.

After efforts for one semester, we have developed an approximate solution for 2L-HFVRP, and implemented software with a graphical user interface for users to use our solution and get the results they need. Our software has been tested with a big quantity of test instances and achieved relatively good results (Please refer to Appendix A - Test report). We also provide a user manual (Please refer to Appendix B).

Since 2L-HFVRP tries to pursue a minimum total distance of all the routes, our solution has the potential to improve economic efficiency by reducing logistics companies’ associated costs and CO2 emissions thus increasing positive externalities for both providers and customers.

Contents

Acknowledgements 3

Abstract 4

List of Figures 6

List of Tables 6

1 Introduction 7

1.1 Customer Needs 7

1.2 Project Tasks 7

1.3 Outline 8

2 Project Aims 8

2.1 Algorithm 8

2.2 Software 9

2.3 Team Building 9

3 Approaches 9

3.1 Process 9

3.2 Terms and definitions 12

3.3 Mathematical Statement 13

3.4 Technology and tools 14

3.5 Algorithm developmen 15

4 Results 17

4.1 Solution to the problem 17

4.2 Software with GUI 18

4.3 Personal Contribution 20

5 Future improvements 20

6 Lesson learnt 20

6.1 What we discovered 20

6.2 What I learnt 21

6.3 Peer-review feedbacks and corrections 21

7 Conclusions 21

References 22

Appendix A: Test report 23

Appendix B: User manual 26

# List of Figures

3.1 Flow chart………………..……………….………………………10

3.2 Project Gantt chart……………………….……………………….11

3.3 Example of the 2L-HFVRP…………….…………………………13

3.4 Technology and tools………………….………………………….15

3.5 Pseudo code..……………………………………………………...16

3.6 UML Diagram…..…………………………………………………17

4.1 Main frame’s display after choosing an input instance……………18

4.2 Result frame……………………………………………………....19

4.3 Main frame’s display corresponding to result frame………..…….19

# List of Tables

Table 1 Milestones……………………………………………………....11

Table 2 Technology and Tools……………………………...……..…….14

# 1 Introduction

Our project is to solve 2L-HFVRP problems (the heterogeneous fleet vehicle routing problems with two-dimensional loading constraints). It is to use different types of vehicles to deliver items to customers. Each vehicle is required to start and terminate at a central depot, and the transported items carried by each vehicle need to be feasibly packed into the loading surface of the vehicle.

## 1.1 Customer Needs

The 2L-HFVRP can be found in many fields related to the transportation of goods, which is an important task in modern society. In the logistics industry a large amount of money is spent daily on fuel, vehicle maintenance and driver wages. In addition to financial costs, the time cost of transportation is also important. Both of them depend on the total distance of transportation routes. Therefore it is important to reduce the total distance of routes when a logistics company needs to deliver a fixed amount of items to customers. By reducing the total distance, the CO2 emissions are also reduced. To improve efficiency and reduce costs, logistics companies expect to have optimized routes to arrange their deliveries. For example, large supermarkets may use 2L-HFVRP solutions to arrange cargo distribution. These actual customer needs drive the research on 2L-HFVRP.

## 1.2 Project Tasks

We are required to solve 2L-HFVRP on the basis of other scholars’ research achievements, and try to improve their solutions by using new packing heuristics algorithm. To achieve this goal, we need to study previous research papers and propose our approach to address 2L-HFVRP. Compared with previous results, we are expected to solve problems of larger size in reduced running time. Moreover, we are also required to deliver a business case study report, a detailed project plan as well as a final report.

Another part of our task is to build software with a GUI to present our solution. A detailed test report and a software user manual would also be provided as attachments of the software.

## 1.3 Outline

The remainder of this report is laid out as follows. In section 2 I describe, in detail, what our project aimed to achieve. In section 3 I describe the process and tools we used to achieve these goals, as well as several algorithms and heuristics we adopted in our project. In section 4, I discuss what we achieved, including our final solution, the results of test cases and evaluation of our software, as well as my personal contribution in the project. Finally, in section 5, 6 and 7, I discuss possible future improvements and lessons I have learnt, and conclude with a summary of the project outcomes.

# 2 Project Aims

## 2.1 Algorithm

Our project intends to find an approximate solution that could minimize the total distance of all routes for 2L-HFVRP.

## 2.2 Software

We aim to develop software with a graphical user interface to present our solution. The software is able to read an input instance, including possible vehicle types, locations of depots and customers, as well as each customer’s items to be delivered. After calculation, the software needs to propose a set of approximately optimal routes in graphical mode.

## 2.3 Team Building

We, group 9, aim to cooperate closely with each other and allocate tasks evenly to each member. Moreover, our team aims to make a detailed project schedule and try to stick to it. We intend to hold regular meetings with our supervisor plus internal meetings within the team, and maintain necessary documentations. Furthermore, we plan to perform peer-reviews within our team as a quality – control measure.

# 3 Approaches

## 3.1 Process

### 3.1.1Project flow chart

The whole process of this project is reflected in below Flow Chart. Generally, the process consists of 4 stages: Designing and planning, algorithm and software development, experiments and testing, and finally the maintaining. We have 3 different types of tests in the plan: algorithm verifying, software testing and computational experiments. Besides, we also prepare detailed Risk Management Plan, Communication Plan and Test Plan for the project.[7]

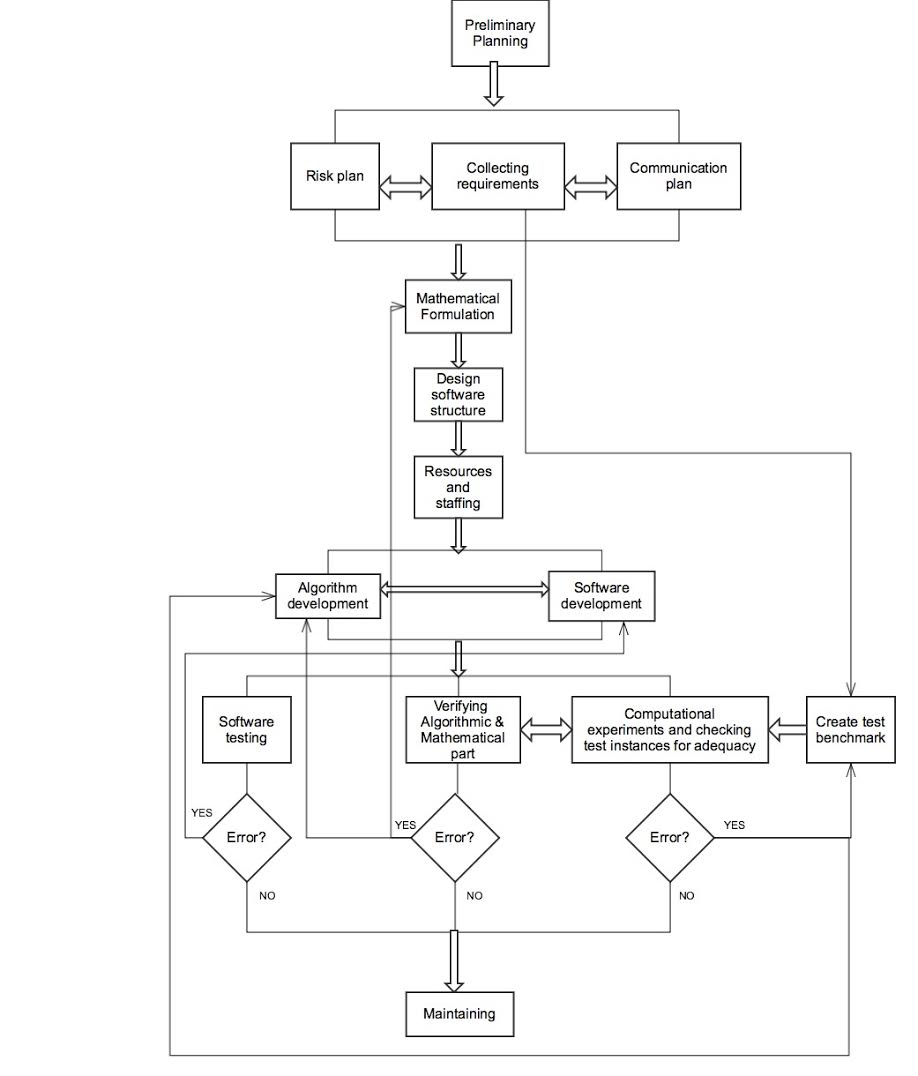


Figure 3.1 Flow Chart

### 3.1.2 Project Schedule

As shown in the Gantt chart below, we intend to finish the project preparation work in early April. Designing, resource allocating and staffing are required to be accomplished in April. The following development for both algorithm and software is planned to commence on 12 April and finish on 20 May, occupying more than 5 weeks. Overlapped with development, the diverse testing work is scheduled to last from early May to early June. Finally, the maintaining work is planned to commence in early June.

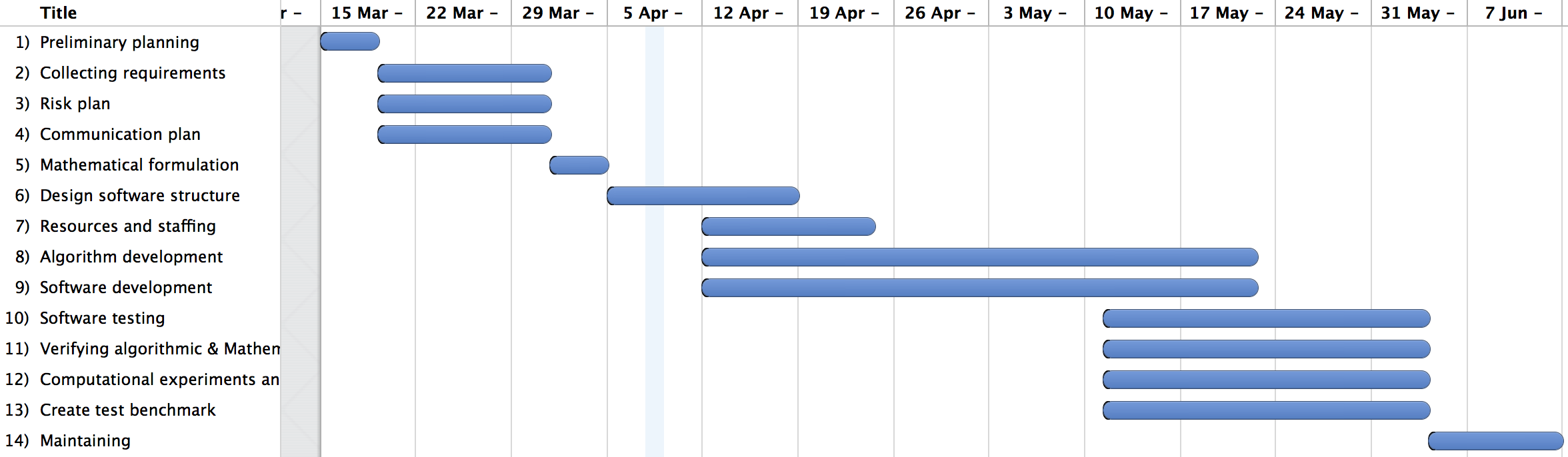


Figure 3.2 Project Gantt chart

We set 5 milestones in the project, as shown below:

|  |  |
| --- | --- |
| Milestones | Reason for setting it |
| Finish requirements collection, mathematical formulation and software structure design | The end of preparation work |
| Finish algorithm development and algorithm verifying. | Ensure the algorithm is prepared well for the next stage. |
| Finish software development | The end of software implementation |
| Finish computational experiments | Verify the correctness of our solution |
| Finish the GUI functionality test | Ensure the GUI works correctly |

Table 1 Milestones

### 3.1.3 Changes in process

The completion of development for both algorithm and software was delayed for about one week, because the algorithm part was more difficult than we thought. After this part, we made efforts to catch up with the schedule and finished subsequent tasks on time.

In terms of the algorithm, initially we intended to consider sequential loading and rotation of items, but we discarded these constraints due to the time limit. These constraints could be the future enhancements to this project. Instead, we changed vehicles from same type to heterogeneous types, which made the problem a little more difficult.

We also adjusted the layout of the GUI per the feedback of peer-reviews from classmates.

## 3.2 Terms and definitions

[2L-HFVRP]: Two-dimensional loading capacitated vehicle routing problem with heterogeneous fleet.

[Depot]: The warehouse. Point 0 in Figure 3.3.

[Customer]: An entity with a fixed location and a fixed amount of items to receive. E.g. Point 1, 2, and 3 in Figure 3.3.  
[Vehicle Type]: A specific type of vehicle with certain size, capacity and fixed / variable cost.

[Route]: The route of a vehicle for delivering items to customers, starting from the depot, serving one or more customers in a fixed sequence and finally ending at the depot. E.g. in Figure 3.3, the route of Vehicle 1 is 0 – 1 – 2 – 3 – 0.

## 3.3 Mathematical Statement

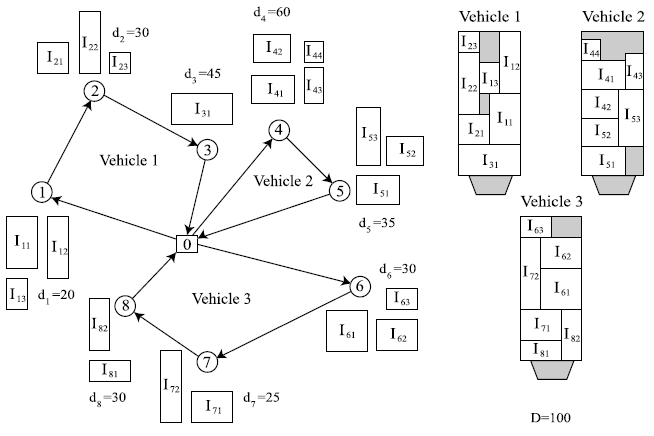


Figure 3.3 Example of the 2L-HFVRP[4]

As shown in Figure 3.3, in 2L-HFVRP, we have different types of vehicles. Each type has capacity Qt, fixed cost Ft, variable cost Vt, loading surface length Lt, and width Wt. Moreover, we have a complete undirected graph G = (V, E), |V| = n + 1, which represents the overall map, and in the graph, each edge eij∈ E has an associated distance dij. Traveling cost of eij by vehicle type t is Cijt = Vt \* dij, because different vehicle types have different variable costs. We add the fixed cost and traveling cost of the vehicle to calculate the total cost of a route. Therefore, the transportation cost of a route for vehicle type t is CR= Ft + \*dR(i), R(i+1),where R is the route whose start point and end point is the depot. Furthermore, we have several customers. Each customer has a demand of mi items of total weight Di. Each item Iir, (r = 1…mi) has a size (wir, lir) and a fixed orientation. We also denote ai=wir \* lir as the total area of the items of customer i. [6] A feasible loading must satisfy the following constraints:

(i) All items of a given customer must be loaded on the same vehicle and split deliveries are not allowed.

(ii) All items must have a fixed orientation and must be loaded with their sides parallel to the sides of the loading surface.

(iii) Each customer can only be served once.

(iv) The capacity, length and width of the vehicle cannot be exceeded.

(v) No two items can overlap in the same route. [6]

The objective of 2L-HFVRP is to assign customer i (i = 1, 2, . . ., n) to one of the routes, so that the total transportation distance / cost are minimized and all the routes fulfill the constraints.[6] As a NP-Hard problem, the real minimum total cost is hard to get. Therefore, we need to find a set of routes, and the sum of each route’s cost should be as small as possible. This means, in an approximate solution, the total cost of all routes should approximate the real minimum value.

## 3.4 Technology and tools

In this project, we follow TDD – Test-driven development process and use the JAVA programming language, SVN as the repository for code, IBM ILOG CPLEX for solving mixed integer problems which helped the packing heuristics, and Trello, which is an online project management tool. Another third-party tool adopted in the project is Linkern.exe, which is used to calculate TSP (Travelling Salesman Problem) route. Each route in 2L-HFVRP is a TSP route since it starts and terminates at the same point – depot.

|  |  |
| --- | --- |
| Technology or Tool | Why we choose it |
| TDD | To ensure that our outcome fulfills the requirements |
| Java | Required by the supervisor |
| SVN | Revision control |
| IBM ILOG CPLEX | Employed in our algorithm part |
| Trello | For online project management |
| Linkern.exe | To calculate TSP route |

Table 2 Technology and Tools



Figure 3.4 Technology and Tools

## 3.5 Algorithm development

Basically we use a multi-agent system for VRP (vehicle routing) part and bin packing heuristics for checking packing feasibility.[2][5] Besides, A linear programming relaxation of a 0-1 integer program is utilized when calculating the initial vehicle set in the beginning.

Firstly, the program reads input files and creates two arrayLists “customers” and “vehicleTypes”, which store the input data. Some calculations are done in this stage. Secondly, an initial set of vehicles is created. Then, according to certain constraints, the program chooses one customer for one vehicle in the initial vehicle set, and creates the first batch of routes. Each vehicle represents one route and each route will work as an independent agent. Thirdly, for each route, it tries to add a customer in arrayList “customers” one after another, and calculates the cost increase assuming this particular customer is added. Subsequently, each route sorts the feasible customers into an ascending sequence based on cost increases, and checks the loading feasibility by packing heuristics for each feasible customer. After that, the route may send an invitation to a specific customer who is feasible and has the smallest cost increase for this route. The actual cost increase is included in the invitation. Finally, each customer who receives at least one invitation from one or more routes could choose the invitation with the smallest cost increase, and join that route. Meanwhile, this customer is removed from the arrayList “customers”.

This process may iterate several times until the program declares that it is infeasible to serve these customers, which means failure, or there is no customer who has not joined a route, which means success.

The GUI module cooperates with this process and displays the result on the screen accordingly.

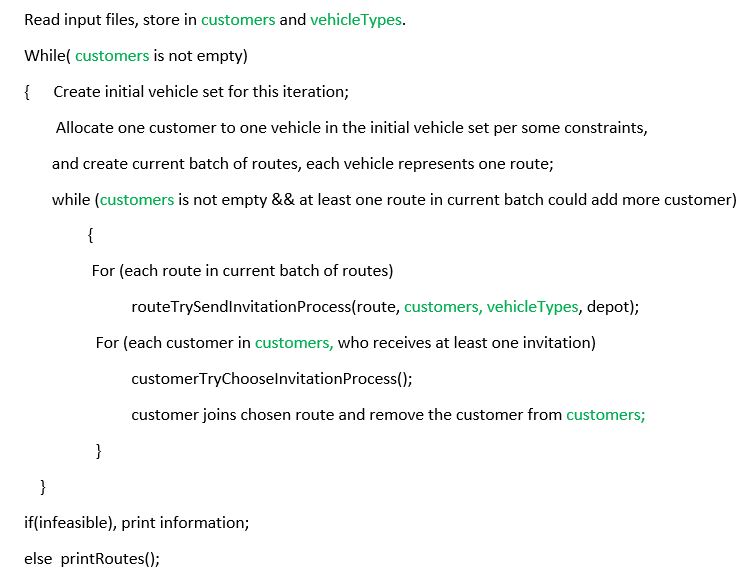


Figure 3.5 Pseudo Code

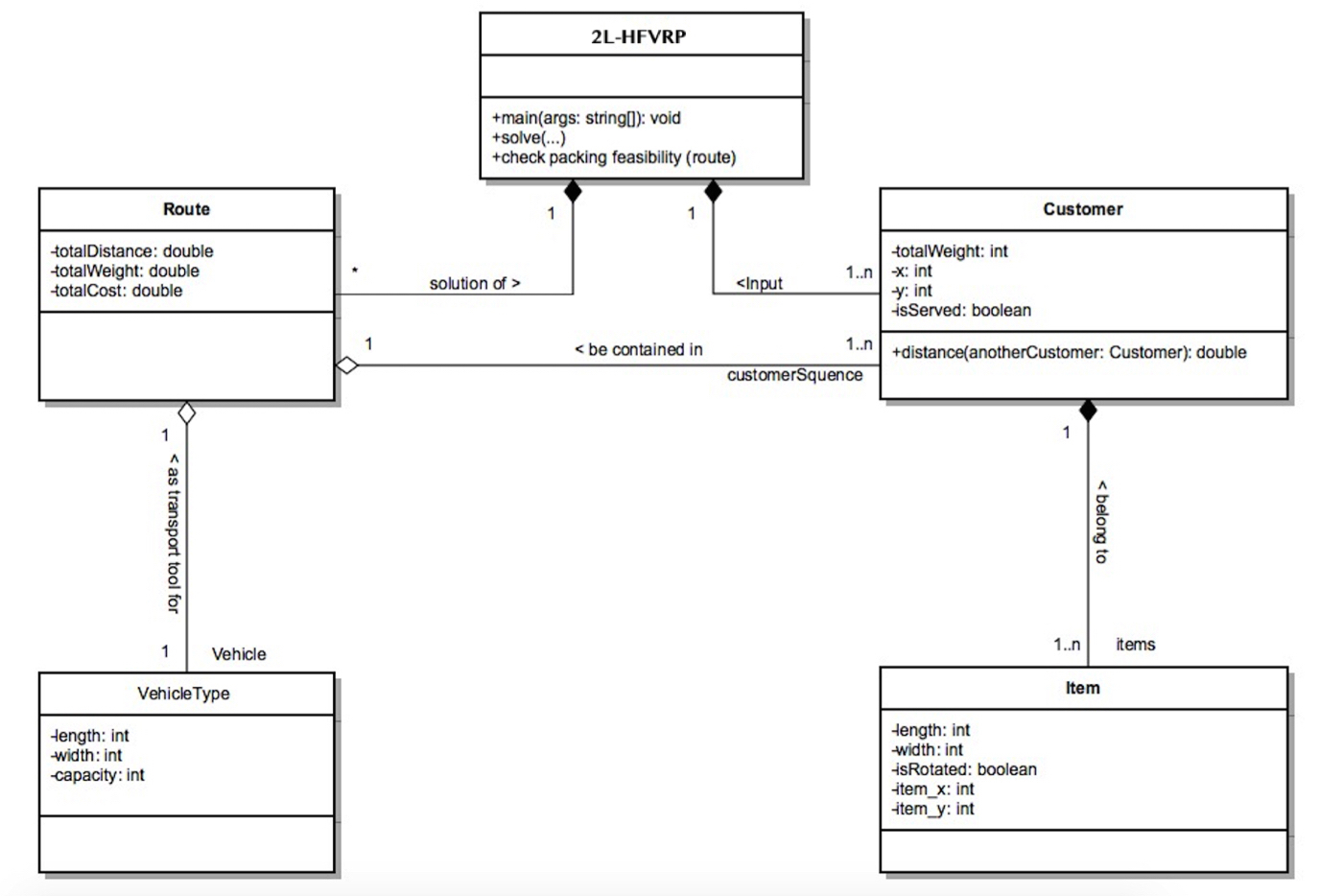


Figure 3.6 UML Diagram

# 4 Results

## 4.1 Solution to the problem

We have achieved an approximate solution for 2L-HFVRP. Our solution is capable to cope with test instances ranging from 15 items for 15 customers to 786 items for 255 customers within acceptable time. It could generate stable and reliable results. Moreover, passing 360 test instances designed by other researchers in this field has proved the quality of our solution. Please refer to Appendix A - Test report.

## 4.2 Software with GUI

We have developed software with a GUI to utilize our solution and presented results in a graphical mode for users. To make it easy to use, we designed a simple interface and provided a software user manual (See Appendix B). With the software, users could browse and select an input file. Then related information of this input instance, including the map, vehicle type as well as detailed description for each customer, will immediately be seen on the screen. As Figure 4.1 shows, in the map, the black hollow square is the depot and other blue points are customers.

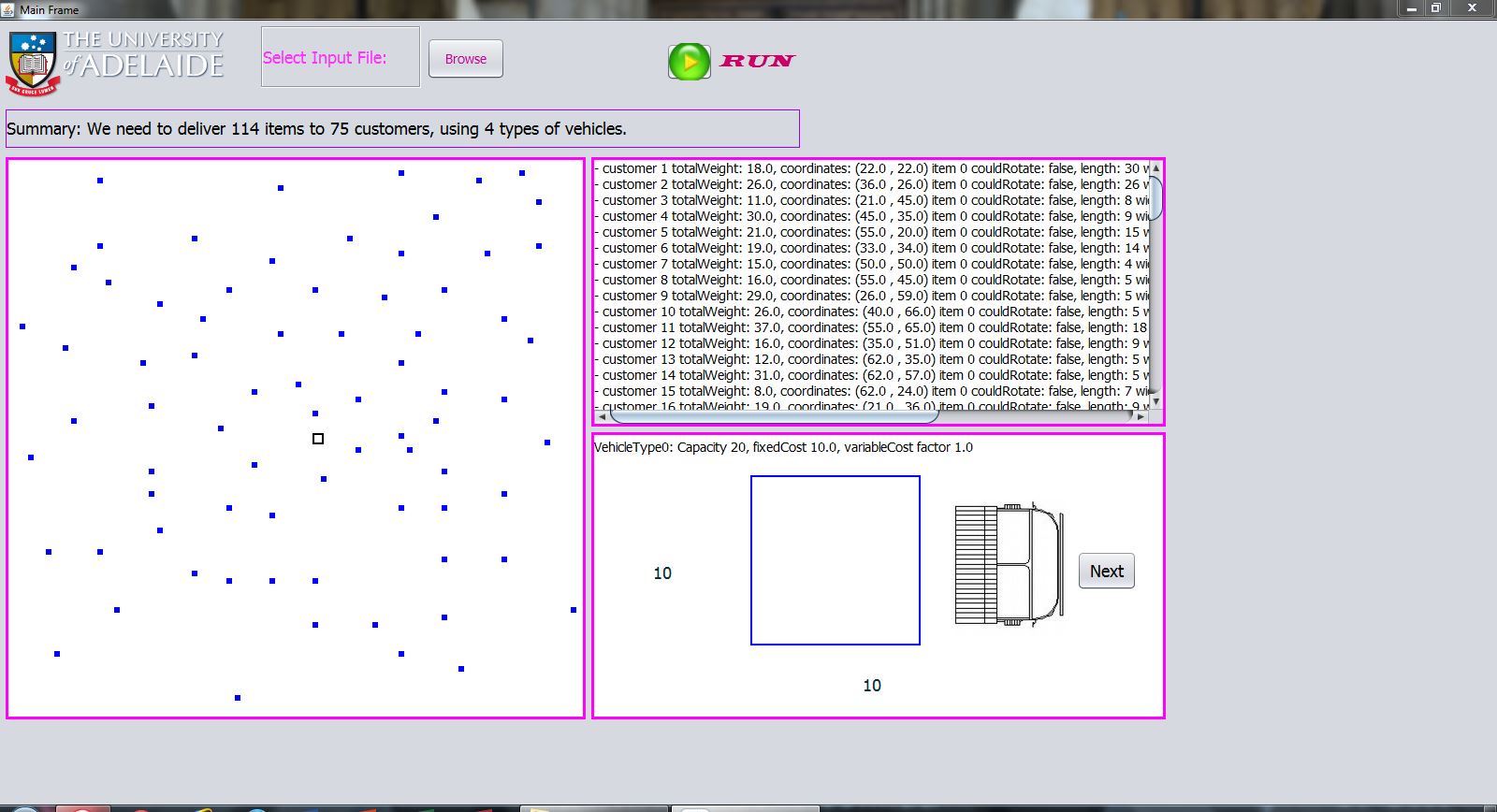


Figure 4.1 Main frame’s display after choosing an input instance

Then press the “Run” button and wait for a while (from 1 second to several minutes depending on the size of input instance) to see the result. The result is displayed in another frame – result frame, which works interactively with the main frame. The result frame mainly presents the route and the items loading plan for that route. It also shows necessary statistics such as number of routes, total distance and total cost. Users could see every route by pressing the “next” button. The display of the main frame corresponds to the result frame in real time by setting the customer points on the current route to red and increasing the size of these points, on the overall map.

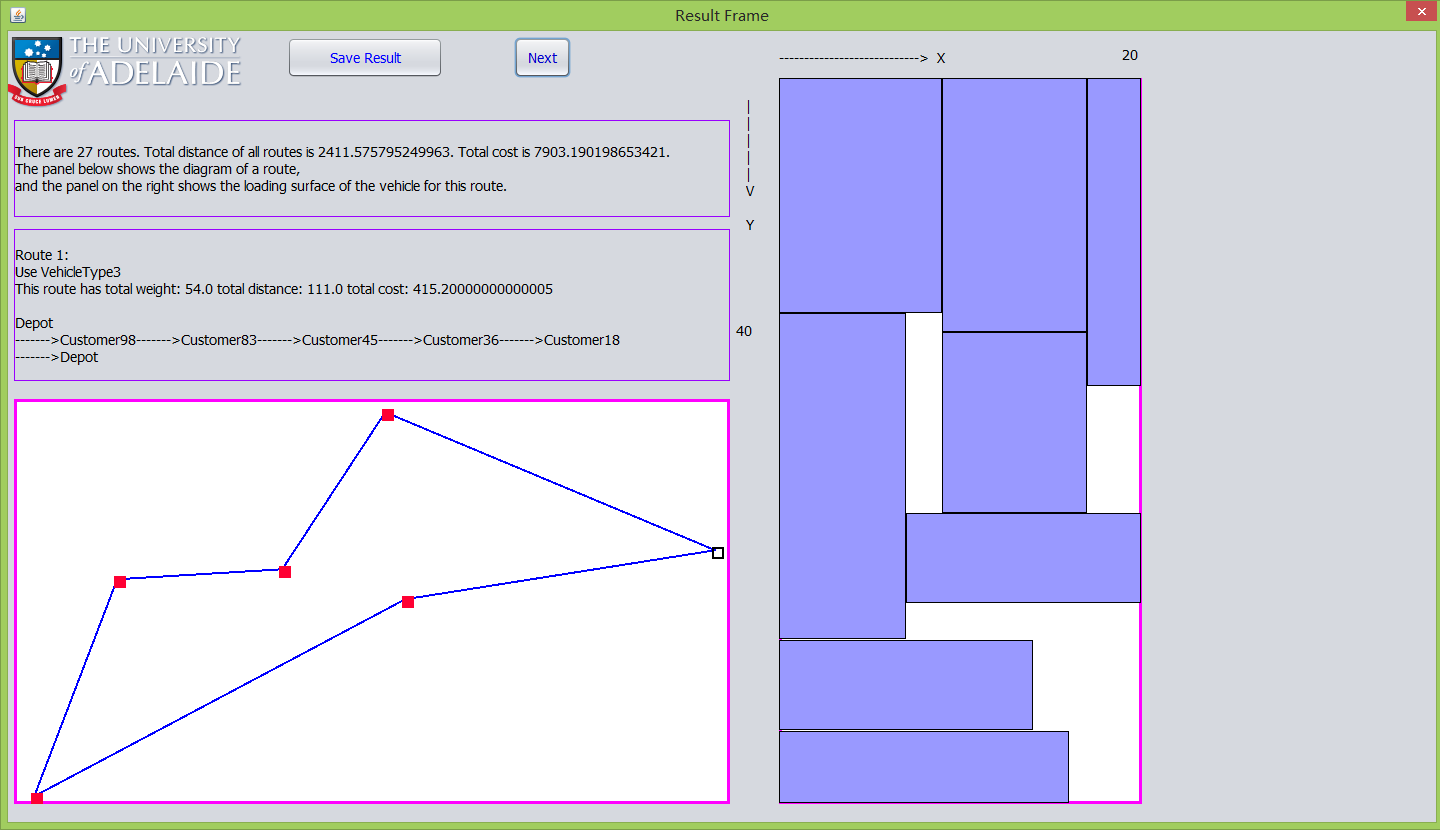


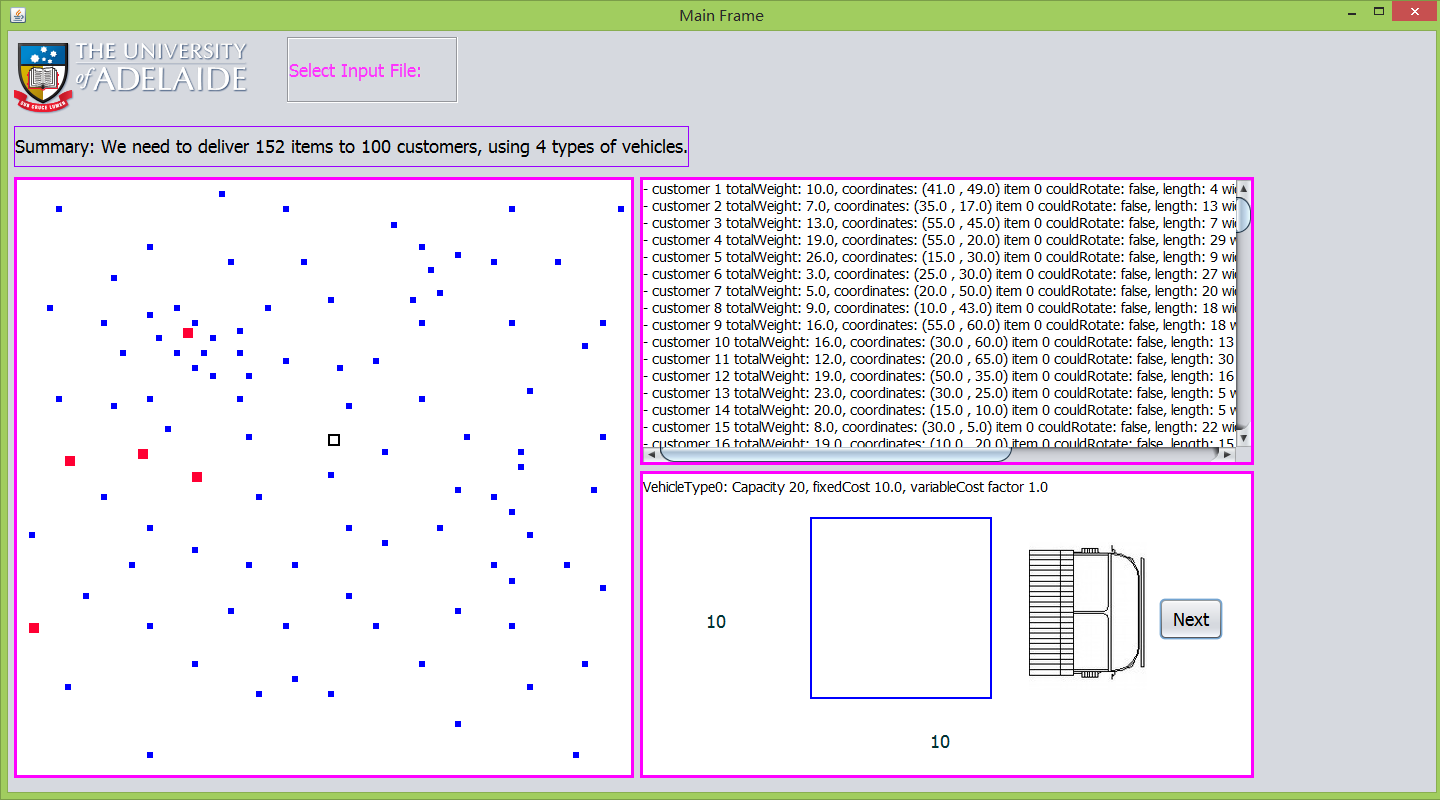
Figure 4.2 Result frame  


Figure 4.3 Main frame’s display corresponding to result frame

In both frames when users move the mouse cursor onto a customer point, depot, or an item in the loading plan, the tip text will show up automatically, which includes the customer identity and coordinates in the map, the item identity and coordinates in the loading surface of the vehicle.

Users could also save the result to a text file in hard disk with the “save result” button.

## 4.3 Personal Contribution

In the process of conducting the project, my contribution firstly includes completing mathematical formulation and designing software structure. Secondly, under the valuable instructions from Dr Sergey and the help of my group members, I managed to complete the main part of algorithm development and software development, and participate in part of the documentation work and tests. To be more specific, I programmed the main part of the software, including 70% of the algorithm part and 90% of the GUI part.

**5 Future improvements**

Due to time limits, sequential loading and item rotation are ignored in our solution. For future improvements, these constraints will be taken into consideration to make the solution more adaptable. In addition, to make the solution more applicable in the real world, researchers could extend this project to use the real road distance between 2 customers instead of the Euclidean distance.

Furthermore, our next plan is to select some real-world logistics companies to try our software, and make improvements according to their feedbacks.

**6 Lesson learnt**

## 6.1 What we discovered

Through debugging and testing, we found that for 2L-HFVRP solutions, the time complexity is mainly determined by the checking packing feasibility subroutine, which is the packing heuristics in our case. Therefore, developing faster packing heuristics or reducing the frequency of calling it, could be a reasonable method to reduce the running time of the whole program.

**6.2 What I learnt**

Through the project, I have realized that software design is much more important than implementation. If the design is not appropriate, the implementation may become more complicated and the outcome may not be satisfactory. Moreover, I have gained much experience about how to work in team, and useful skills in algorithm and software development.

**6.3 Peer-review feedbacks and corrections**

There were 2 concerns in the peer-review section. Firstly, some classmates found it hard to understand the algorithm. The correction we provided was the pseudo code of the whole program and each subroutine. Secondly, when the peer-review was held, our program was run under Linux terminal mode. Most classmates felt it inconvenient to read the result in the Linux terminal. Therefore, we developed a GUI for our program to display the result clearly.

**7 Conclusions**

In the 2L-HFVRP project, our group has obtained an approximate solution and developed software with a GUI to utilize our solution. Both the solution and the software have passed a great number of tests with various input instances. Therefore, the reliability and quality of them could be guaranteed.

**References**

[1] Michel G et al, 2007, A Tabu Search Heuristic for the Vehicle Routing Problem with Two-Dimensional Loading Constraints

[2] Polyakovsky S & M’Hallah R, 2013, ‘A muti-agent system for the weighted earliness tardiness parallel machine problem’

[3] Dominguez O, et al, 2014, ‘Using biased randomization for solving the two-dimensional loading vehicle routing problem with heterogeneous fleet’, DOI 10.1007/s10479-014-1551-4

[4] Iori, M., & Martello, S, 2010, ‘Routing problems with loading constraints’, TOP, 18, 4–27

[5] Polyakovsky S & M’Hallah R, 2007 ‘An agent-base approach to the two-dimensional guillotine bin packing problem’, European Journal of Operational Research, 192,767-781

[6] Leung S, et al, 2013, ‘A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints’

[7] Zhao X, Luo Y & Zhang W, 2015, ‘Project Plan of Group 9’, 2015 MCI project course, School of Computer Science of the University of Adelaide.

**Appendix A: Test Report**

1. **Algorithm verifying**

For each subroutine or function which belongs to the algorithm part, we wrote specific codes to test them one by one. Basically, we created special and typical test inputs and printed out the related information and results in the terminal.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Description** | **Comment** | **Decision** |
| 1-1 | Function. getInitialNumberOfVehicles() | Work correctly after 3 times modification | Accept |
| 1-2 | Function. initiaRoutes() | Work correctly | Accept |
| 1-3 | Function. checkScale() | Work correctly | Accept |
| 1-4 | Function. calculateDelta() | Work correctly | Accept |
| 1-5 | Function. pass\_packingHeauristic() | Work correctly | Accept |
| 1-6 | Function.resetRoute() | Work correctly | Accept |
| 1-7 | Function.sendInvitation() | Work correctly | Accept |
| 1-8 | Function.linkernTour() | Work correctly after 10 times modification | Accept |
| **Test procedure** |  |  |  |
| **Step number** | **Operator actions** | **Expected result and evaluation criteria** | **Result** |
| 1 | Create special test inputs | / | OK |
| 2 | Run the function in main() | / | OK |
| 3 | Compare the output with expected output | / | OK |

1. **Computational experiments:**

The test instances are provided by other researchers in this field. We obtained 360 test instances from them, ranging from 15 items for 15 customers to 786 items for 255 customers. Part of the recorded results and running time is reflected here:



1. **GUI functionality test**

We tested all functionalities of the GUI one after another.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test ID** | **Description** | **Comment** | **Decision** |
| 3-1 | Log in module test | Work correctly | Accept |
| 3-2 | Mainframe “Browse” button test | Work correctly | Accept |
| 3-3 | Mainframe display test | Work correctly | Accept |
| 3-4 | Mainframe “Run” button test | Work correctly | Accept |
| 3-5 | Resultframe display test | Work correctly | Accept |
| 3-6 | The consistency of mainframe and resultframe display | Work correctly | Accept |
| 3-7 | Result save module test | Work correctly | Accept |
| **Test procedure** |  |  |  |
| **Step number** | **Operator actions** | **Expected result and evaluation criteria** | **Result** |
| 1 | Run the software | / | OK |
| 2 | Test the specific functionality | / | OK |
| 3 | Verify the behavior of the GUI | / | OK |

**Appendix B: Software User Manual**

**1. Intended users:**

Logistics companies and / or logistics departments in enterprises.

**2. Software Summary:**

This software could read an input instance, which includes vehicle types, locations of depot and customers, as well as each customer’s items to be delivered. After calculation, the software could propose a set of approximately optimal routes in a graphical mode.

**3. Environment:**

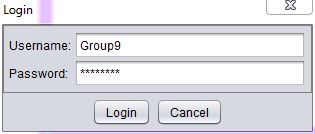
Window 7, Window 8/8.1

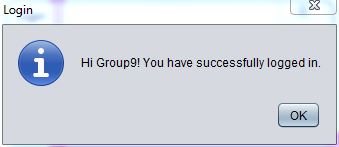
**4. Instructions:**

4.1. Double-click the “Group9.exe” executable file or desktop shortcut icon:

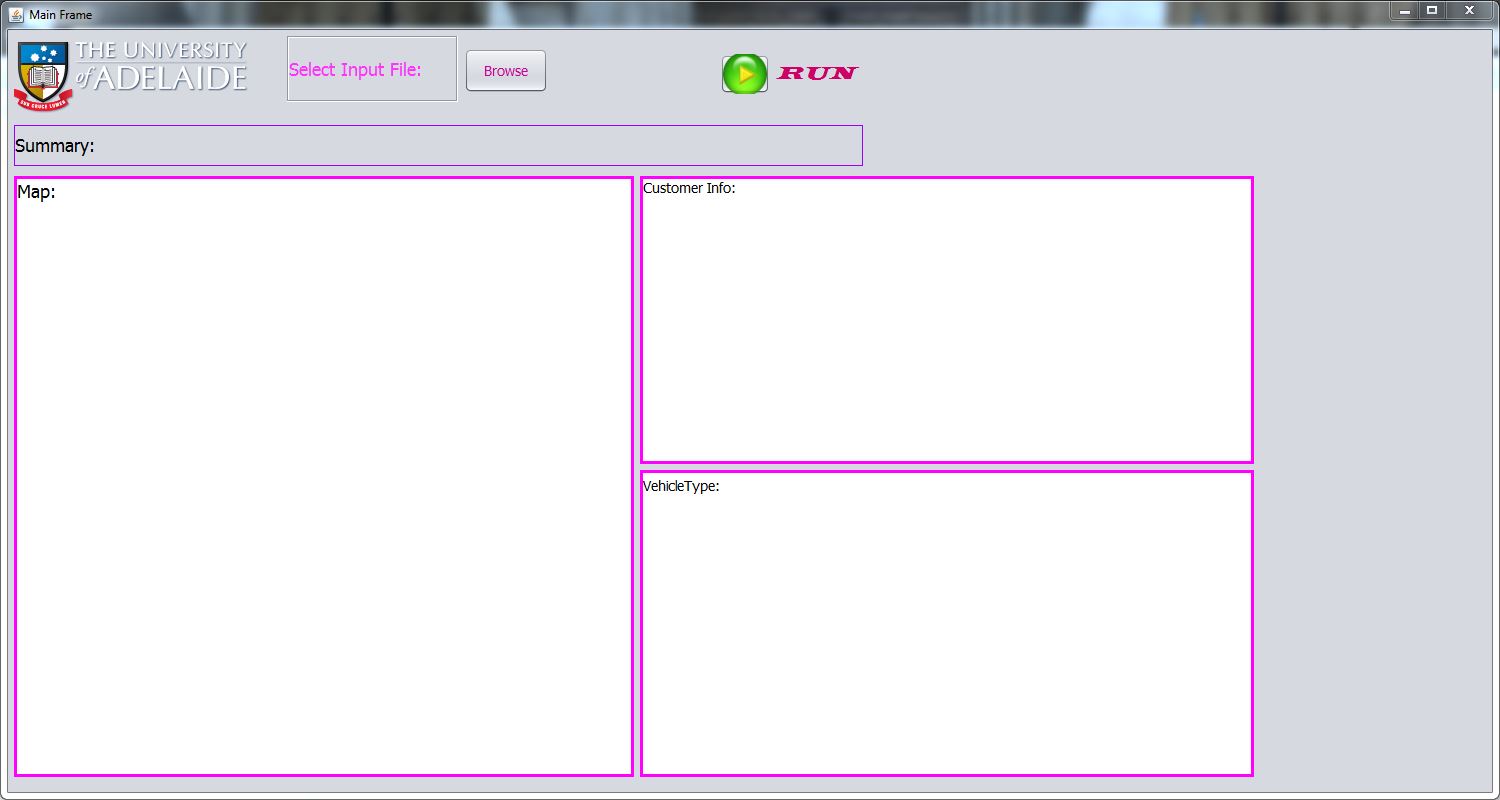


4.2. A log in dialog will show up and users need to log in with his/her account and password:

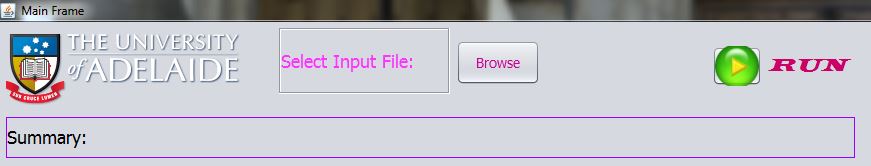


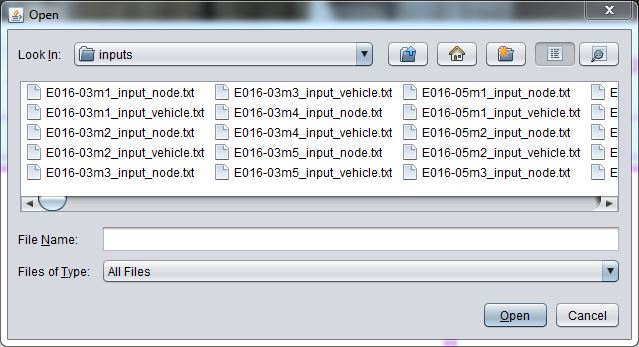


4.3. After log in successfully, a user is able to operate on the main Frame : There are two buttons, “Browse” is for selecting an input instance, and “Run” is for starting the calculation and subsequently showing the result.

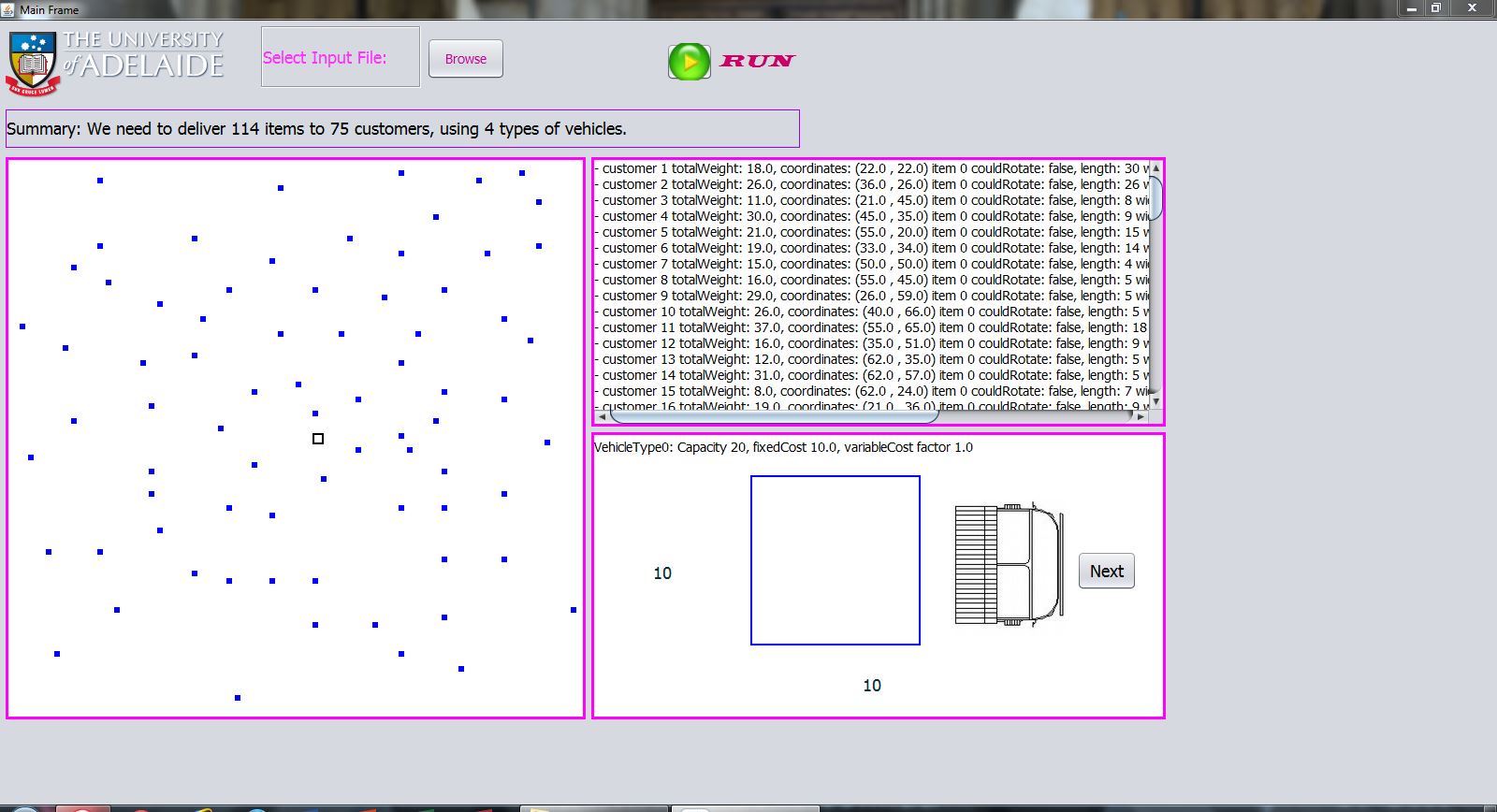


4.3.1 Click “Browse” to select an input file:



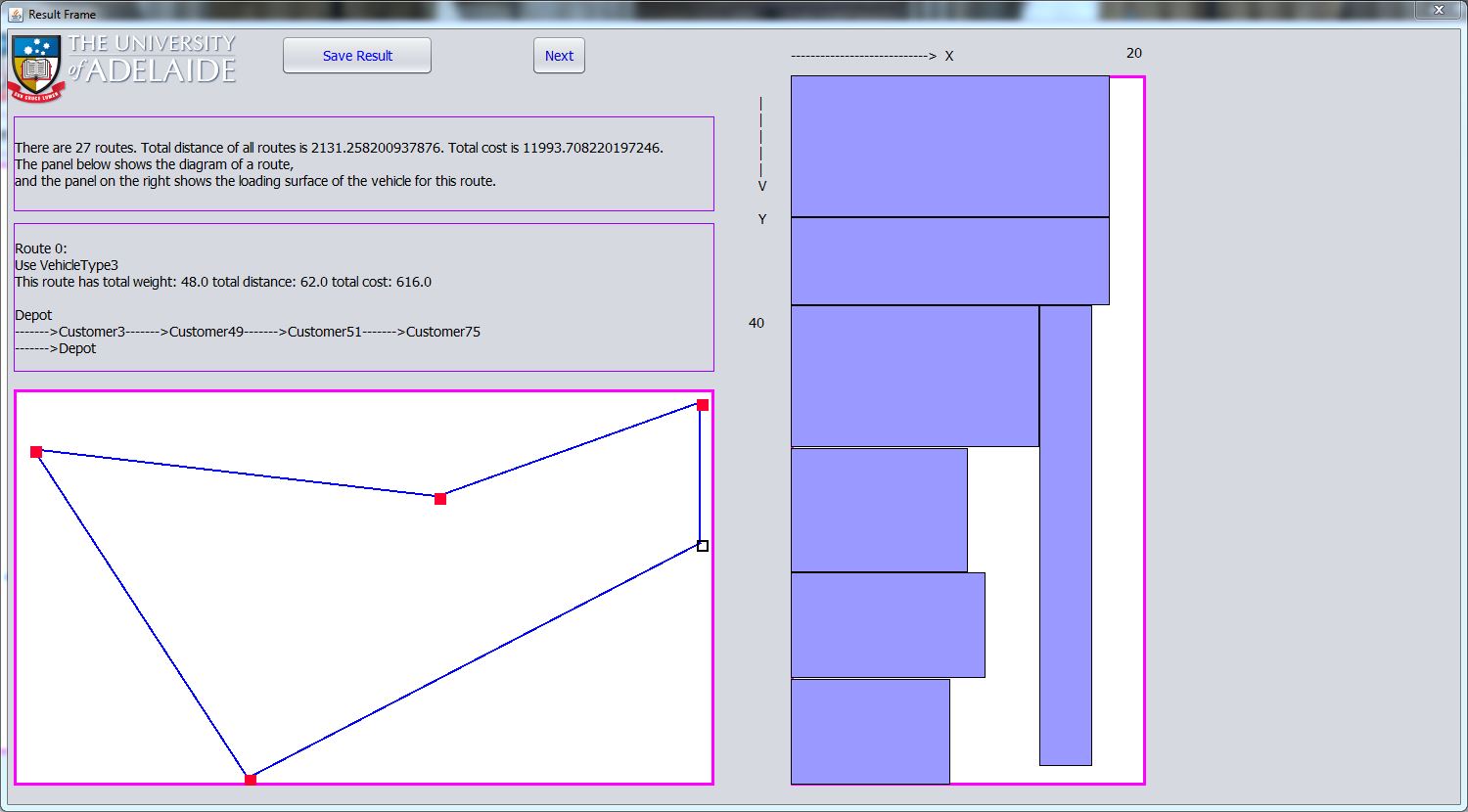


When a user selects an input file, the related information, including the map, vehicle type as well as detailed description for each customer, will immediately be seen on the screen. In the map, the black hollow square is the depot and other blue points are customers.

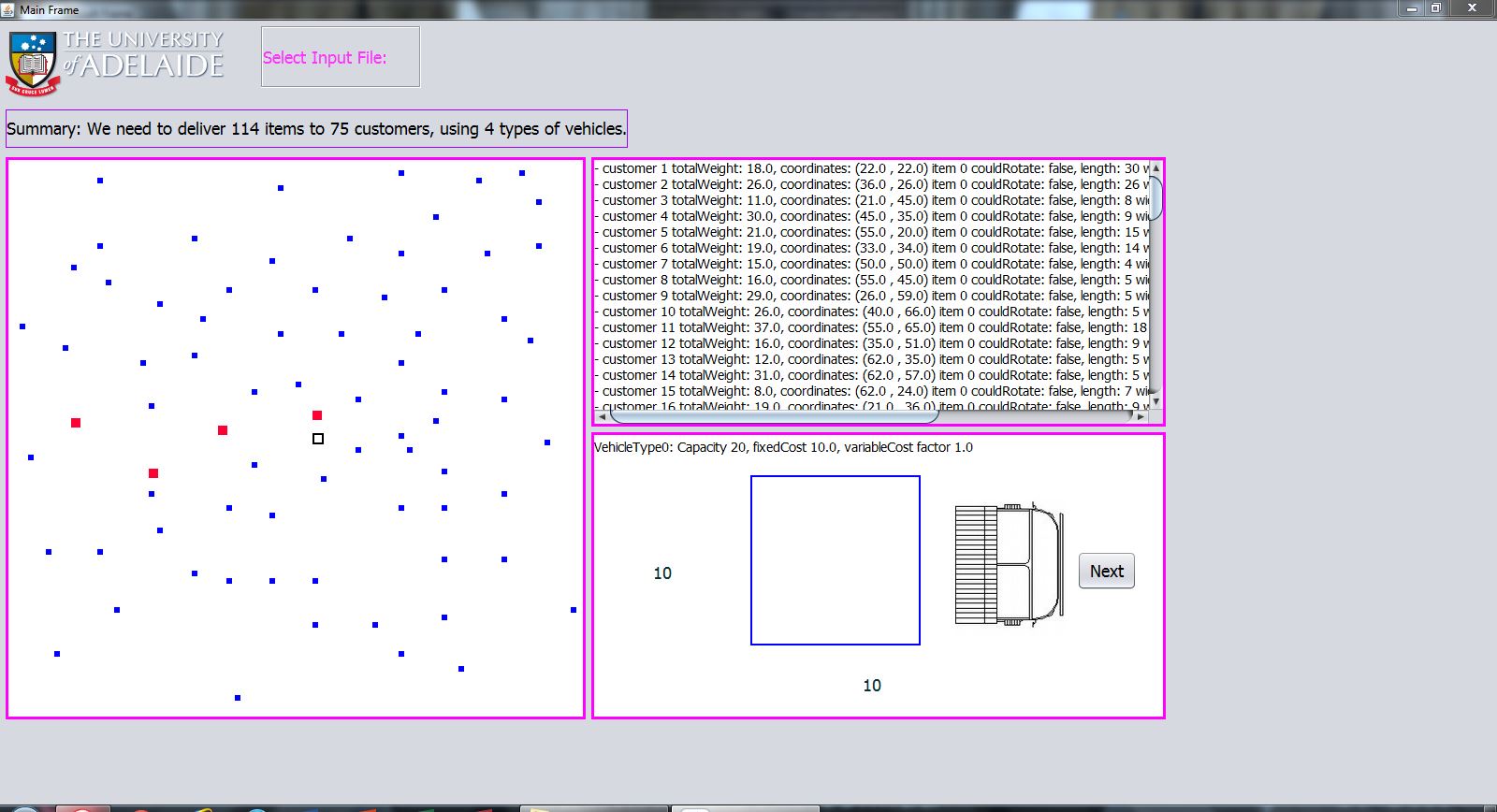


Main frame display after selecting an input file

4.3.2 Then press “Run” button and wait for a while (from 1 second to several minutes depending on the size of the input instance) to see the result. The result is displayed in another frame – result frame, which mainly presents the route and items loading diagram for that route. It also shows necessary statistics such as number of routes, total distance and total cost. The user could see every route by pressing “next” button. The display of Main Frame is corresponding to Result Frame in real time by setting the customer points on the current route to red and increasing the size of these points, on the overall map.



Result frame typical display



The main frame display corresponding to the result frame display

4.3.3 In both frames when users move mouse cursor onto customer points, depot, or items in loading surface, the tip text will show up automatically, which includes the customer identity and coordinates in the map, the item identity and coordinates in loading surface of the vehicle.

4.3.4. Save results. Another button on result frame is the “save result” button. By pressing this button users could save the result to a text file in hard disk.

