



香港城市大學
City University
of Hong Kong

Department of Electronic Engineering

FINAL YEAR PROJECT REPORT

BScIT-2002/03-CS/HWC-201

**Vehicle Routing Problem Using
Ant Colony Optimization**

Student Name: Wang Yingpei

Student ID: 50095232

Supervisor: Dr. CHUN, Andy H W

Assessor: Dr. Chan, Sammy C H

Bachelor of Science (Honors) in
Information Technology

Acknowledgements

I would like to express my deepest gratitude to thank my supervisor Dr. Andy Chun, my assessor Dr. Sammy Chan for providing me the greatest support and constructive comments in accomplishing this project.

First of all, I would like to thank my supervisor Dr. Andy Chun, who has provided fantastically thoughtful, comprehensive, well-organized information and feedbacks throughout the whole project with great patience and enlightening guide. Also, I would like to thank my assessor Dr Sammy Chan for his invaluable support to this project.

Finally, thank all my friends for their great encouragement and concern.

Wang Yingpei, Anita

April 23, 2003

TABLE OF CONTENTS

<i>Content</i>	<i>Page</i>
Acknowledgement.....	II
List of Figures	V
List of Tables.....	VI
Abstract.....	1
Chapter 1: Introduction.....	2
Chapter 2: Background	3
2.1: VRPTW.....	3
2.1.1: VRP.....	3
2.1.2: VRPTW.....	3
2.2: Ant Colony System.....	5
2.2.1: Real Ants.....	5
2.2.2: Artificial Ants.....	6
Chapter 3: Implementation.....	10
3.1: ACS Model for VRPTW.....	10
3.1.1: Multi-Objective ACS.....	10
3.1.2: Relationship with TSP.....	11
3.2: Data Definition.....	12
3.3: Main Structure.....	13
3.3.1: Flow Chart.....	13
3.3.2: Nearest Neighbor Heuristic.....	13
3.4: Ant Colony to Minimize Vehicle.....	16
3.4.1: Unfeasible Solution.....	16
3.4.2: Insertion.....	16
3.4.2.1: Selection.....	17
3.4.2.2: My Modification.....	18
3.4.2.3: Insertion Details.....	19
3.4.3: Structure of ACS_Vehicle.....	25
3.4.3.1: Building Model.....	25
3.4.3.2: Details of the ACS_Vehicle.....	26
3.5: Ant Colony to Minimize Time.....	29
3.6: Adaptation to Real Life Application.....	31
3.6.1: Limitation.....	31
3.6.2: HK Geographical Characteristic.....	31
3.6.3: Modification of Building Model.....	32
3.6.4: Change in the Visibility Function.....	33
Chapter 4: Experiment Result.....	34
4.1: Data Set.....	34

4.2: Parameter Setting.....	34
4.3: Interface Design.....	35
4.4: Testing Result.....	36
4.4.1: Clustered Data.....	36
4.4.2: Random Data.....	36
4.4.3: Comparison with parameter changes.....	37
4.4.4: Training process of ACS.....	37
4.4.5: Analysis.....	38
Chapter 5: Conclusion.....	40
5.1: My Contribution.....	40
5.2: Discussion About VRPTW and ACS.....	41
5.3: Future Development.....	42
Reference.....	43
Appendix 1: Sample Result of Traditional Method.....	44
Appendix 2: Sample Result of Reloading Method.....	47
Appendix 3: NP-Complete Problem.....	50

LIST OF FIGURES

<i>Number</i>	<i>Page</i>
1. Traditional VRP	3
2. VRPTW	5
3. Ant Colony Concept.....	6
4. Traveling Salesman Problem.....	7
5. 2-objective ACS.....	11
6. Building Model.....	12
7. Flow Chart of Main Structure.....	15
8. Infeasible Solution.....	17
9. Insertion Sample.....	18
10. Insertion Selection.....	20
11. First Category.....	21
12. Second Category.....	21
13. ...17 Time Constraint 1.....	24
18. ...22 Time Constraint 2.....	25
23. Allow Reloading Model.....	30
24. Interface Design.....	35
25. Clustered Data.....	36
26. Random Data.....	36
27. Table of Parameter Comparison.....	37
28. Training Process.....	37

LIST OF EQATION AND PSEUDO CODE

<i>Number</i>	<i>Page</i>
1. Equation (2.1)(2.2)(2.3).....	8
2. Equation (2.4)(2.5).....	9
3. Equation (3.1)(3.2)(3.3)(3.4).....	13
4. Equation (3.5)(3.6)(3.7).....	14
5. Equation(3.8)(3.9)(3.10)(3.11).....	19
6. Equation (3.12)(3.13)(3.14).....	28
7. Equation (3.15).....	33
8. Pseudo Code 1.....	10
9. Pseudo Code 2.....	16
10. Pseudo Code 3.....	27
11. Pseudo Code 4.....	29
12. Pseudo Code 5.....	30

Vehicle Routing Problem Using Ant Colony Optimization

ABSTRACT

The Vehicle Routing Problem with Time Windows (VRPTW) involves scheduling and routing of a vehicle fleet to serve a given set of geographically distributed requests with capacity and time constraints. This problem is encountered in a variety of industrial and service applications. Specific examples include bank deliveries, postal deliveries, and school bus routing and scheduling.

In my project, the algorithm, Ant Colony Optimization (ACO) is used to solve VRPTW. The aim is to investigate and analyze the performance of ACO in the VRPTW. Moreover, extra features are added to adapt to HK geographic characteristic, a more clustered and compact area. The features allow the fleet to return to the depot for reloading in the middle of the route, which is only suitable in small and clustered area.

The project is coded by C#. The test of traditional ACO is based on the converted Benchmark problem [Solomon 1987] while the test of adapted ACO is based on realistic data set which is more literally in real life. The results of traditional ACO and adapted one will be analyzed. An interface is added to demonstrate the fleet route and its training process.

Chapter 1

Introduction

Transportation is an important fraction of the economical, social and environmental aspects. It is the response to the ever-growing needs for contacts between individuals, companies or societies. Therefore, the internal structure and distribution management of the transportation systems are very important.

We view the problem as a combined vehicle routing and scheduling problem which often arises in many real-world applications. However, there are 2 real life considerations:

1. Man power are costly, it is a common objective for companies to reduce man powers. Therefore, the purposes of the vehicle routing problem are to reduce the number of fleet and minimize the total delivery time.
2. Vehicle routing problem with time windows are an NP- hard problem. The manual solution of this problem is very time-consuming. Besides, the time scheduling is a routine task for many companies, each day there should be a new solution according to different data set. Therefore, the computer solution to vehicle routing problem is essential to increase the efficiency of daily work.

Hong Kong has a geographical characteristic of highly clustered and small. Some modifications are applied to the vehicle routing problem to see whether it is more suitable in Hong Kong. Originally, when each vehicle finishes the capacity of its load, it will return to the depot and its work finish. In Hong Kong, since the depot is near the destination customers, we can allow the vehicle to reload at the depot and serve more customers.

Chapter 2

Background

2.1 Vehicle Routing Problem with Time Windows (VRPTW)

2.1.1 Vehicle Routing Problem (VRP)

A set of Clients with known demands must be served by a fixed fleet of vehicles of limited capacity. All vehicles are located at a common depot. The vehicle routing problem (VRP) is to serve each client exactly once, so as to minimize the total route length. The customers' demand can not be split, and the total demand on one route can not exceed the capacity of the vehicle. Refer to **Figure 1**.

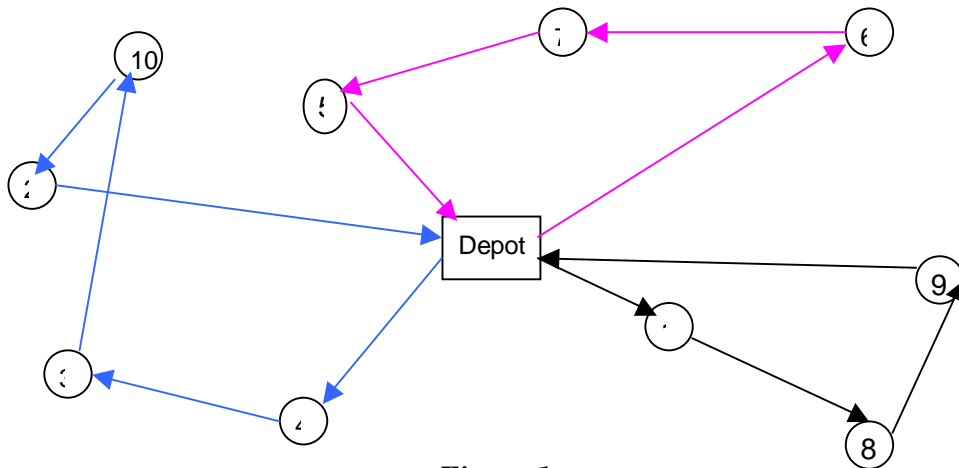


Figure 1

2.1.2 Vehicle Routing Problem with Time Window (VRPTW)

Developed from VRP, VRPTW add time constraints to the customer and depot to achieve real life application. This problem includes:

- For the depot and for each customer c_i ($i=0, \dots, n$), there is a time window $[b_i, e_i]$ during which the customer has to be served (with b_0 and e_0 be the time constraints for depot)

- b_i – the earliest time the customer can be served
- e_i – the latest time the customer can be served
- s_i – each customer involves a service time
- The time window can no be violated, the vehicle should arrive between $[b_i, e_i]$.
If the vehicle arrives earlier, it has to wait till the b_i . But it can never arrive after the e_i .
- Same as VRP, each customer has a demand which can not be split
- Each vehicle has a capacity limit.

The major objective is to minimize the number of fleet due to real life situation that driver salaries are variable costs for the company or when the company has to rent vehicles to perform deliveries.

Other optimization objectives are: minimize the total delivery time, minimize the total waiting time (vehicles can arrive earlier than time window and have to wait). **Figure 2** is a brief illustration of the VRPTW.

There are total 3 vehicles in the graph, each one depart from the depot and return to the depot. The customer's time window can not be violated. The whole demand of the customer on one route can not exceed the vehicle's capacity.

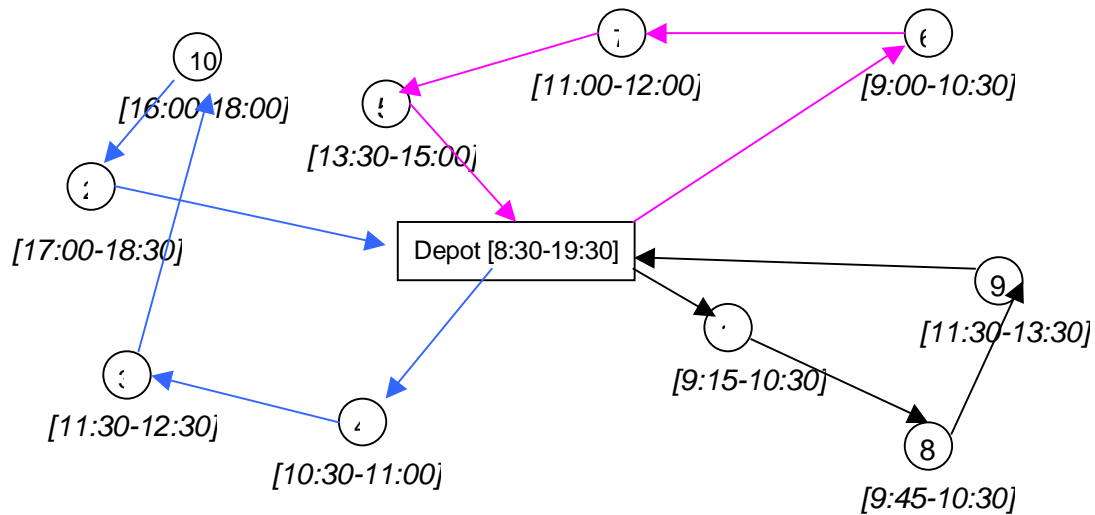


Figure 2

2.2 Ant Colony System

This section will introduce the swarm intelligence – Ant Colony System and its application on Traveling Salesman Problem (TSP). TSP is a basic model for VRPTW.

They are of great similarities.

2.2.1 Real Ants

Real ants are capable of finding the shortest path from a food source to the nest (Beckers, Deneubourg and Goss, 1992; Goss, Aron, Deneubourg and Pasteels, 1989) without using visual cues (Hölldobler and Wilson, 1990). Refer to **Figure 3**. Ants are moving on a straight line that connects a food source to their nest. It is well known that the primary means for ants to form and maintain the line is a pheromone trail. Ants deposit a certain amount of pheromone while walking, and each ant prefers to follow a direction rich in pheromone. Obviously, within the same period of time and suppose the ants choose route with same probability at first, the shorter route will have more ants after a while and the pheromone on that route is stronger than that on

the longer route. Therefore, the next choice of ants will bias which lead to more and more ants choose the shorter route. (Ants will return to the home and retransmit).

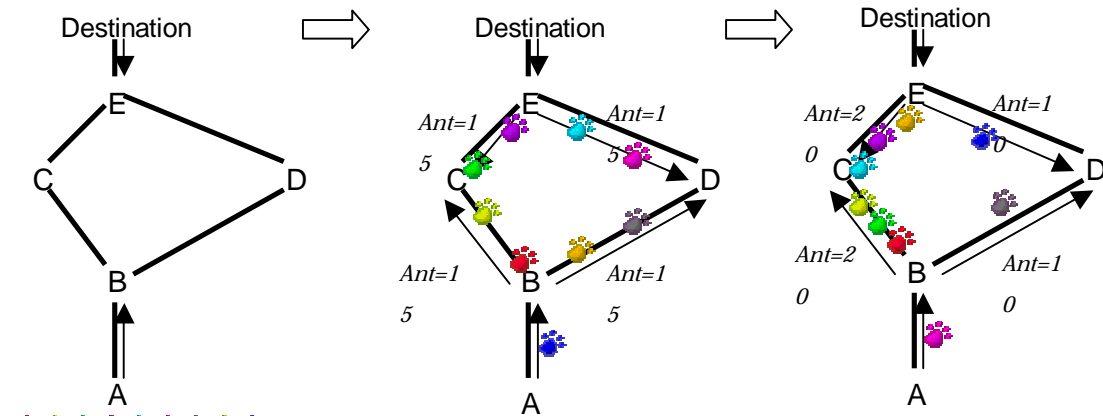


Figure 3

2.2.2 Artificial ant

Take Traveling Salesman Problem (TSP) as an example (very useful in the later discussion of VRPTW). TSP is a problem that one salesman has to visit all the cities and find the shortest path. All the cities are fully connected in this example.

An artificial ant is an agent which moves from city to city on a TSP graph. It chooses the city to move to using a probabilistic function both of trail accumulated on edges and of a heuristic value, which was chosen here to be a function of the edges length. Artificial ants probabilistically prefer cities that are connected by edges with a lot of pheromone trail and which are close-by. Initially, m artificial ants are placed on randomly selected cities. At each time step they move to new cities and modify the pheromone trail on the edges used –this is called local trail updating. When all the ants have completed, select the ant that uses the shortest tour and modifies the edges belonging to its tour. It is called global trail updating, by adding an amount of

pheromone trail that is inversely proportional to the tour length. These are three ideas from natural ant behavior that we have transferred to our artificial ant colony:

- The preference for paths with a high pheromone level
- The higher rate of growth of the amount of pheromone on shorter paths
- The trail communication among ants.

Artificial ants can determine how far away cities are, and they have a working memory, a TABU list, used to memorize cities already visited (the working memory is emptied at the beginning of each new tour, and is updated after each time step by adding the new visited city).

Refer to **Figure 4** for model of TSP

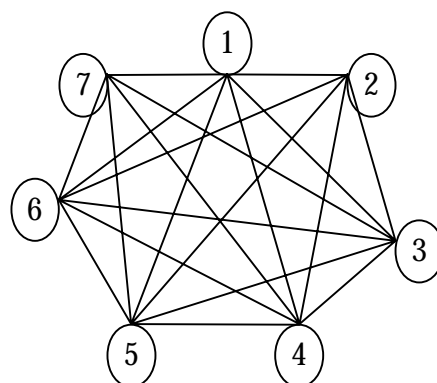


Figure 4

Notation:

d_{ij} – distance from city i to city j . The distance can be calculated by x,y coordination or directly given by user.

τ_{ij} – pheromone on the edge of city i to city j

η_{ij} – visibility of city j from city i . it is the inverse of d_{ij} , which shows the closeness of city j to city i .

$p_{ij}^k(t)$ – probability of ant to choose city j when it is on city i at time t . while j is a feasible candidate city of ant k .

Each artificial ant acts as a salesman. They have a memory, a buffer which will record the sequence of the visited cities, called Tabu List. These artificial ants are randomly placed on the cities, which are the first cities in the Tabu list of these ants.

The ant will choose next city based on this equation:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^\alpha \cdot [\eta_{ik}]^\beta} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases} \quad (2.1)$$

where α and β are two adjustable parameters that control the relative weight of trail intensity, τ and η .

To encourage exploitation and prevent early convergence, there is an improved method.

An ant on city i choose the city j to move according to this rule:

$$j = \begin{cases} \arg \max u \in J_i \{ [\tau_{ij}(t)] \cdot [\eta_{ij}] \} & \text{if } q \leq q_0 \\ J & \text{if } q > q_0 \end{cases} \quad (2.2)$$

While J is the city selected by equation (2.1)

After each ant finishes its route, calculate the total route length of each ant. If an improved solution is found, then replace the temporary global best route.

Then for every edge in the global temporary best route (may not be changed if no improved solution has been found), update the pheromone of that edge:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta \tau_{ij} \quad (2.3)$$

Where $\Delta \tau_{ij} = 1/L^+$. L^+ is the route length of global temporary best route.

Since each time only the best route will be updated pheromone, after several iteration, it will accumulate more and more pheromone. All the ants are likely to converge to that best route while exploration of new routes has little hope. Therefore an improved method arises to reduce the attraction of the best route. Before the iteration has finished, each ant will perform local pheromone update if it go through that route. The local update rule will reduce that edge's pheromone. If all the ants go through the best route, then the pheromone on that best route will reduce a bit, which provide new chances for other route. The local update route is:

$$\tau_{ij}(t) = (1 - \rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_0 \quad (2.4)$$

$$\text{Where } \tau_0 = \frac{1}{(n \cdot L_{nn})} \quad (2.5)$$

L_{nn} is the route length of the result generated by Nearest Neighbor Heuristic¹. From the equation, τ_0 is much smaller than τ_{ij} . So the pheromone will decrease during iteration to reduce attraction.

Here is the pseudo code for the basic TSP problem:

¹ Nearest Neighbor Heuristic(NHH): The NHH builds a solution starting from a randomly selected city and repeatedly selecting the nearest unvisited city as the next city until all cities have been visited.

```

/* Initialization*/
Initialize n ants
Calculate a feasible route using Nearest Neighbor Heuristic and store it as global
temporary best route
Assign  $\tau_0$  by equation (2.5)
For every edge (i, j) do
     $\tau_{ij}(t) = \tau_0$ 
End For
For k=1 to n do
    Place ant k on a randomly chosen city
End For

/*Main Loop*/
For t = 1 to  $t_{MAX}$  (iteration of Ant Colony)
    For k = 1 to n do
        Construct the route of ant k by equation (2.1) and equation (2.2);
        After each step, do local pheromones update by equation (2.4) and (2.5)
    End For
    For k=1 to n do
        Calculate the route length of ant k, select the shortest
    End For
    If There is an improved result
        Replace the global temporary result
    EndIf
    For every edge (i, j) belong to the global best path
        Do global pheromone update by equation (2.3)
    EndFor
EndFor
/* Parameter Setting*/
 $\rho = 0.1, \beta = 1, \alpha = 1; q_0 = 0.9$ 

```

Pseudo Code 1

Chapter 3

Implementation

3.1 ACS model for VRPTW

3.1.1 Multi-objective ACS

Vehicle Routing Problem is a direct extension of the Traveling Salesman Problem. It requires all the customers be visited exactly once and minimize the total delivery time. The difference is that: in TSP, only one salesman travels all the cities; while in VRP, several vehicles visit all the customers and each vehicle has a capacity limitation. VRPTW is a development from VRP which add more constraints to the customers. Since there are 2 objectives, both of them are optimized simultaneously by coordinating the activities of two ACS based colonies. The goal of the first colony, ACS-VEI, is to try to diminish the number of vehicles used, while the second colony, ACS-TIME optimize the feasible solutions found by ACS-VEI. Both colonies use independent pheromone trails but they share a common global best route.

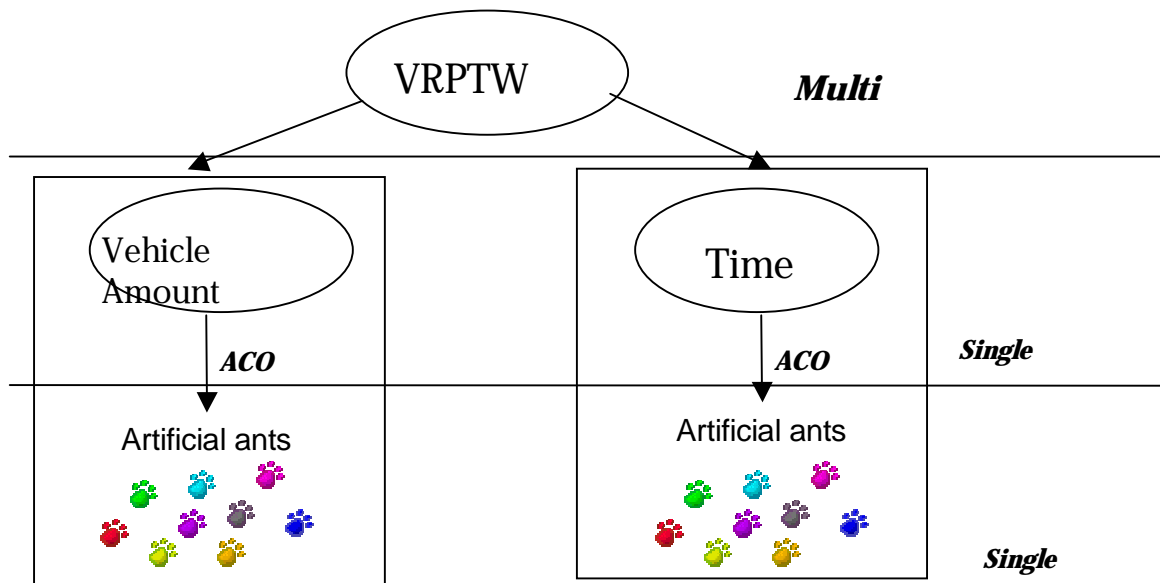


Figure 5

3.1.2 Relationship with TSP

In TSP, only one sales man travels all the cities. Each artificial ant acts as a salesman and they have a Tabu List which exactly records all the cities.

But in VRPTW, several vehicles added up to visit all the customers. One artificial ant is not enough to represent several vehicles. To solve this problem, the depot is duplicated a number of times equal to the number of available vehicles. Distances between copies of the depot are set to zero. The approach makes the VRP closer to the traditional traveling salesman problem. Refer to figure 6.

So, both in the TSP and in this model, we can use only one artificial ant to visits all the nodes exactly once. Duplicated depots (d_0, \dots, d_3) are grey points while clients are white points. All duplicated depots have the same coordinates but they have been split to clarify the picture. Each time the ant return to either one of the depots, it finishes the route of one vehicle. From the map, we can see 4 routes.

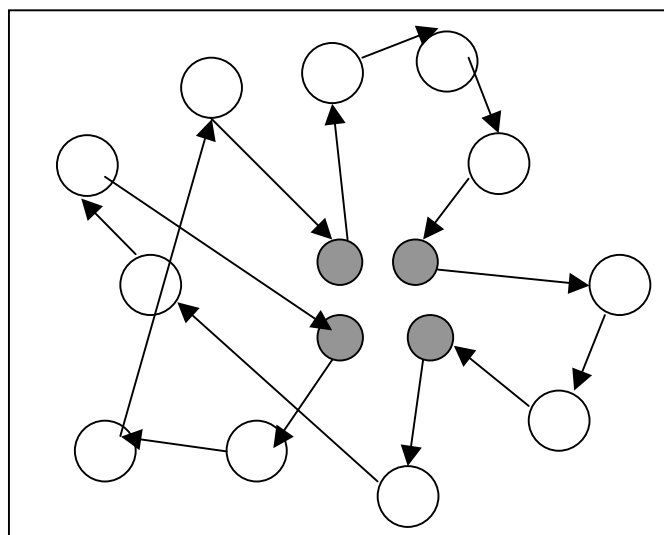


Figure 6

3.2 Data Definition

The VRPTW can be formally stated as follows:

Given a set of customers and a depot, $C = \{1, 2, \dots, n\}$ represents the set of customers to be served while C_0 is the depot. The vehicles are the same with the capacity Q . Every customer $i \in C$ has a positive demand d_i , service time s_i , and a time window $\{b_i, e_i\}$ in

which the service should start. b_i are the earliest time the service can begin, e_i is the latest time the service can begin. x_{ij}^k is one if vehicle k drives from node i to node j , and 0 otherwise. arr_i^k denotes the time for which a vehicle k starts to service customer i .

A cost c_{ij} and a travel time t_{ij} are associated with each edge. The service of the customers must be feasible with respect to the capacity of the vehicles, and the time windows of the customers serviced. The objective is to first minimize the number of routes, and then the total distance of all routes.

Here is the mathematical definition of the constraints in the project:

We try to **MINIMIZE**:

$$\sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij}^k \quad (3.1)$$

While we must **OBEY**

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ij}^k \leq Q \quad (x_{ij}^k \in \text{Route } k) \quad (3.2)$$

Means the total demand on the route is less or equal than the vehicle capacity Q

$$\sum_{j \in N} x_{0j}^k = 1 \quad (x_{ij}^k \in \text{Route } k) \quad (3.3)$$

$$\sum_{i \in N} x_{i, n+1}^k = 1 \quad (x_{ij}^k \in \text{Route } k) \quad (3.4)$$

$$\sum_{i \in N} x_{ih}^k - \sum_{j \in N} x_{hj}^k = 0 \quad \forall h \in C, x_{ij}^k \in \text{Route } k \quad (3.5)$$

These 3 equation means the vehicle must start from the depot and return to depot.

$$arr_i + s_i + t_{ij} \leq b_j \quad \forall i, j \in N \quad (3.6)$$

$$b_i \leq arr_i \leq e_i \quad \forall i \in N \quad (3.7)$$

These 2 equations means the time window can not be violated.

3.3 Main Structure

3.3.1 Flow Chart

The first ACS of vehicle will generate a solution with least number of vehicles. Based on that solution, the ACS of Time will try to find the shortest delivery time using this amount of vehicles. Both of the two colonies are independent. They use different pheromone set and artificial ants. But they share a common global temporary best solution, which will contribute to the global pheromone update in each ant colony.

Refer to **Figure 7** on next page.

3.3.2 Nearest Neighbor Heuristic

The initial solution is generated by Nearest Neighbor Heuristic. All the customers must be visited no matter how much vehicles are used. We use the same mode as in ACS but without pheromone update, the depot is duplicated (see section 3.1.2). Only one ant is used. The ant starts from one depot and select the next customer by calculating the distance. It selects the nearest candidate customer while not violating the time window constraints and capacity constraints. Once there is no feasible customer, it returns to one split of the depot. Then the current time and the vehicle load will be cleared. The vehicle goes out as if it is the beginning of the route.

Refer to **Pseudo code 2** on page 15.

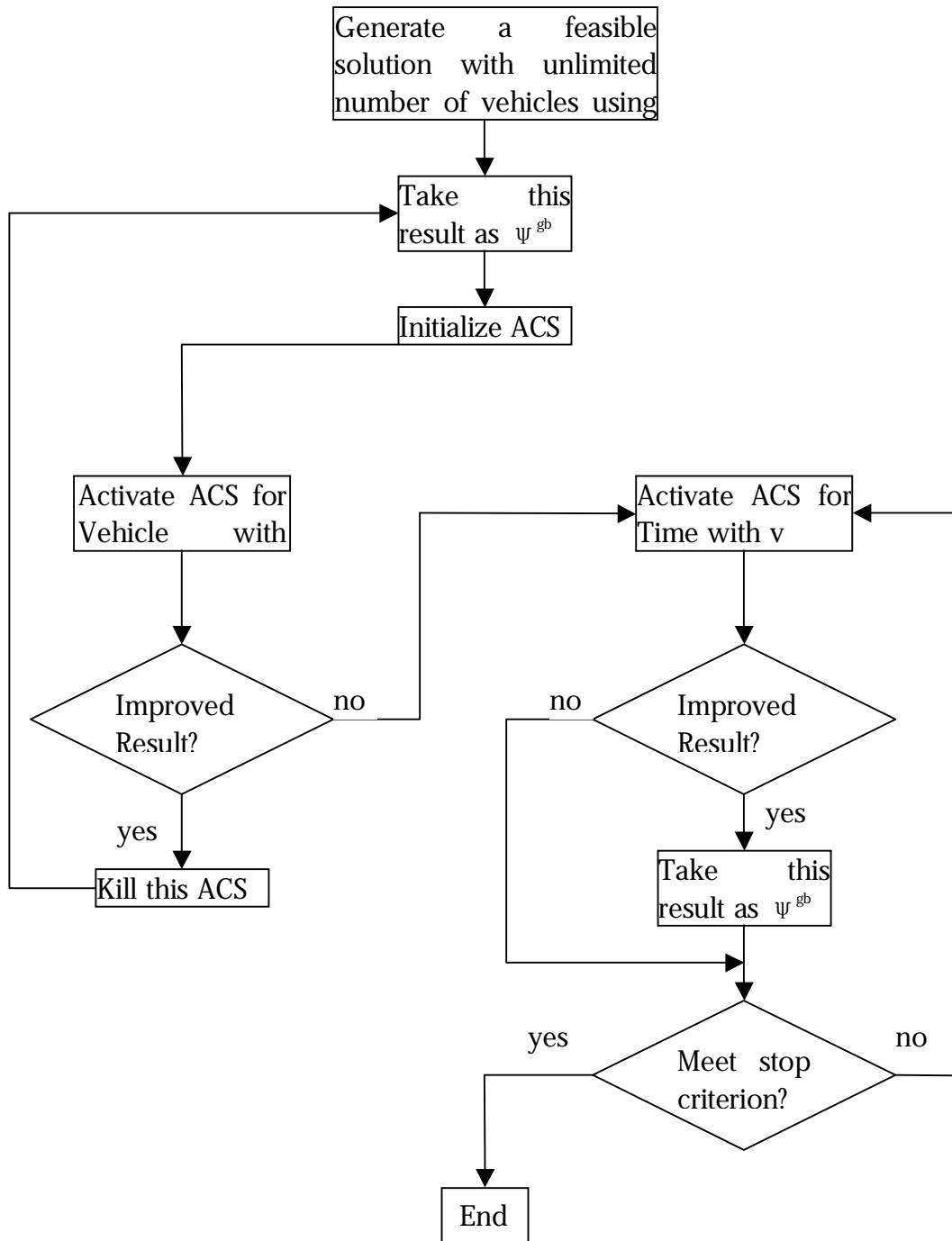


Figure 7

/*Initialize*/ : guess the number of the vehicle may be needed.

No. = average demand of customer * customer number / vehicle capacity

The no should be far bigger than the final result to ensure that a reasonable result can be generated.

/*Main loop*/:

generate one vehicle

FOR the vehicle route has not been fulfilled

FOR from the first customer to see whether it has been visited

IF the vehicle capacity is 0

Then the next customer should be the depot

Every parameter of the vehicle should be initialized

Break;

ENDIF

IF customer demand not exceed capacity and this customer has not been visited

Then calculate the time the vehicle has to travel from current customer(or depot) to this customer (time_span)

IF (time_span + current_time +service_time< begin_time)

New time_span = begin_time - current_time -service_time

ENDIF

IF (time_span + current_time < end_time)

IF time_span< min (select the minimum time span)

Min=time_span

ENDIF

ENDIF

ENDIF

IF V_capacity >0 but it can no more satisfy any unvisited customer

The vehicle has to go back to the depot

ENDIF

ENDFOR

Assign the next customer to the vehicle route and update the current time, vehicle capacity and customer status.

EndFor (until all the customer has been visited)

Report the whole route to the global bestpath

Pseudo Code 2

3.4 The Ant Colony for Minimize Vehicle Number

3.4.1 Infeasible Solution

The any colony tries to generate a feasible solution with $(v-1)$ vehicles, one vehicle less than the smallest number of vehicles for which a feasible solution has been computed.

There remains a problem: The available vehicle may not be able to visit all the customers because of time constraints and capacity constraints.

In Figure 8, there are 3 customers which have not been visited.

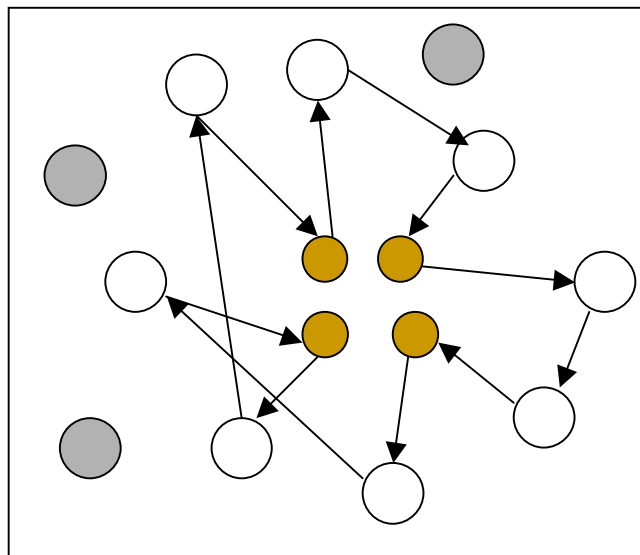


Figure 8

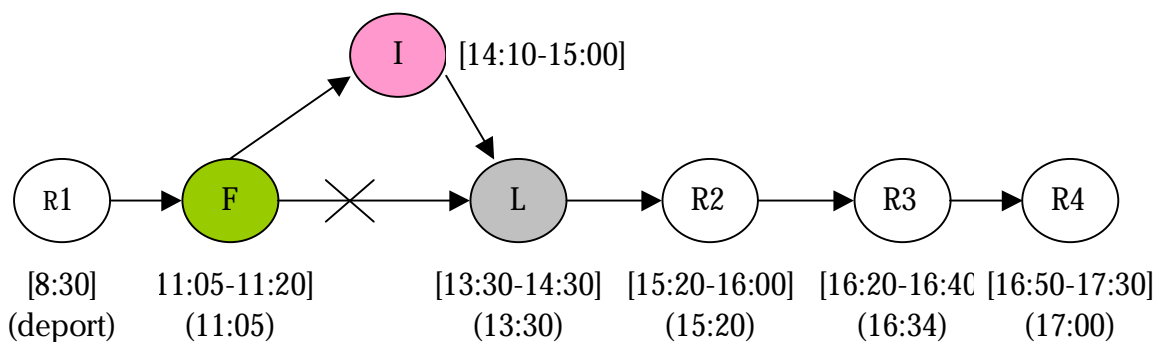
To solve this problem, we have to use insertion. Because in the Ant Colony, each ant will choose the next cities due to a combination factor of distance, transmission time, demand and wait time. In other words, these unvisited customer can be possibly inserted in one of these routes without violate the constraints.

3.4.2 Insertion

The existing insertion method including

- Nearest Insertion – The same as nearest neighbor heuristic
- Farthest Insertion – If partial tour T does not include all the cities, find cities j and k. j is on the tour but k is not on the tour, for which $C(j,k)$ is maximized. (C (,) is the insertion cost)
- Cheapest Insertion – If partial Tour T is not include all the cities, find for each k not on T, the edge (i,j) for T which minimize $C(T,k)=C(i,k)+C(k,j)-C(i,j)$. Select the city k with the minimum $C(T,k)$
- Greedy Insertion – Place all the visited customers in one set, and place all the unvisited in another set. Then select one from unvisited set according to specific criteria and insert in the visited set.

Since our problem has constraints as time window and capacity limit. These heuristics must be modified to be applicable. Refer to **Figure 9**. The time in the () is the actual arriving time. The time in [] is the time window



Figure

3.4.2.1 Selection:

We focus on the sequential insertion heuristic [Solomon 1987]. It is based on cheapest insertion. This heuristic applies two criteria: one for selecting best position of the unvisited customer and the other for the customer who has the best cost. The cheapest insertion cost and the associated insertion place for each unvisited customer are calculated using the following formulas:

$$C1 = d_{iu} + d_{uj} - d_{ij} \quad (3.8)$$

$$C2 = b_{j_{new_begin}} - b_{j_{old_begin}} \quad (3.9)$$

$$C_{total} = \alpha_1 \cdot C1 + \alpha_2 \cdot C2 \quad \text{where } \alpha_1 + \alpha_2 = 1 \quad (3.10)$$

i is the former city, u is the city to be inserted and j is the latter city. C is the cost of insertion.

3.4.2.2 My Modification:

In the implementation of Solomon's method, I find there is another restriction – the customer demand. After insertion, the total delivery amount on that route can not exceed the vehicle's capacity. But according to Solomon's method, some customer with large amount may miss the chances to be inserted first and when it is his turn, none of the route has such a big space. Or the insertion is doomed to be impossible because of demand restriction, and then the previous selection and insertion are useless which waste computation time.

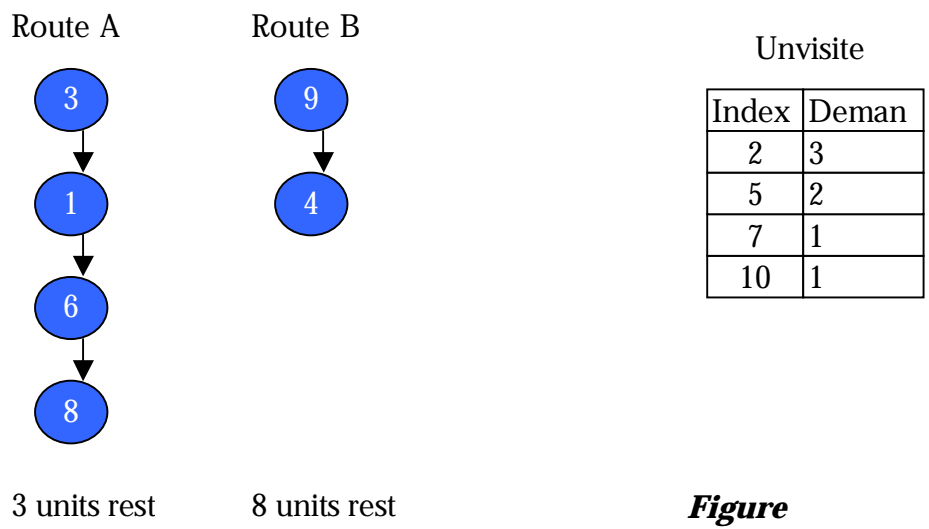
Therefore, I add up each route's rest space and sort the routes. Meanwhile, I also sort the demand of the unvisited customer.

I modified Solomon's equation (3.10) by adding one factor of demand:

$$C_{total} = \alpha_1 \cdot C1 + \alpha_2 \cdot C2 + \alpha_3 \cdot rank \quad \text{where } \alpha_1 + \alpha_2 + \alpha_3 = 1 \quad (3.11)$$

After select the candidate customer, I will check whether any one of the routes can meet such a demand. If none of the route has such a large amount to meet the Max_Demand Customer, then this ant's work failed. Refer to **Figure 10**.

When comes to this step, the next task is to try inserting the customer in Route B. If it fails for other constraints, then try route A. If it fails again, this ant's work failed.



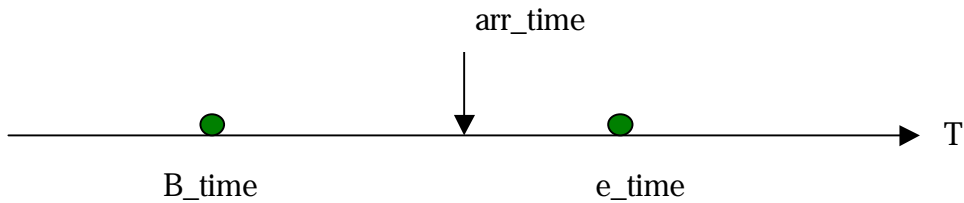
Figure

3.4.3.3 Insertion

After select the candidate insertion customers, the next thing to do is to insert in the existing route. The demand constraints has been solved in the selection phase, so there remains only time constraints.

As indicated before, each customer has a time window $\{b_i, e_i\}$ and if the vehicle arrives earlier, it has to wait till the b_i . So the arr_time of each customer has 2 categories.

1. **First Category:** The arr_time of the latter customer is between b_time and e_time . **Figure 11**



Figure

2. **Second Category:** The *arr_time* is before the *b_time*, so the drive has to wait, then the *arr_time* becomes the *b_time*. **Figure 12**



Figure

Because of these 2 categories, the influence of insertion will be different according to these situations. For explanation simplicity, refer to **Figure 9**, We call the former city **F**, the candidate city **I**, and the latter city **L**. The rest cities after **L** are **R_i**

➤ **For First Category:**

For the L city:

1. If the travel time to new city plus the travel time from new city to L city is longer than the original travel time between city F and L, but arrive at the city L before its end time, refer to **Figure 13**. Therefore the $\Delta \text{time} > 0$, and will influence the later city, which will be discussed later. (the service time is included in the [Time from * to *] for simplicity)

Time from F to I + Time from I to L > Time from F to L & Time arrive L < L
e_time

2. If the travel time to new city plus the travel time from new city to L city is shorter than the original travel time between city F and L, but arrive after the begin time of city L. Refer to **Figure 14**. Therefore the $\Delta \text{time} < 0$,

Time from F to I + Time from I to L < Time from F to L & Time arrive L > L
b_time

3. Based on above situation, but the arrival time at city L is before its begin time, Refer to **Figure 15**. Therefore the $\Delta \text{time} < 0$, but the difference from figure 2 is that the drive has to wait till the b_time of this customer.

➤ **For Second Category:**

For the L city:

1. If Time from F to I + Time from I to L > Time from F to L & The arr_time is after the L's b_time. Refer to **Figure 16**. Therefore the $\Delta \text{time} > 0$.
2. If Time from F to I + Time from I to L < Time from F to L. Then although the $\Delta \text{time} < 0$, but it won't influence the later cities, because the drive has to wait till the b_time. Refer to **Figure 17**.

➤ **For The rest of cities** (suitable for both first and second Category)

If $\Delta \text{time} > 0$:

1. If the arr_time is later than the b_time, then just add the Δtime and check if it exceeds the e_time of this city. If not, then this city is not influenced by the delay. Refer to **Figure 18**.
2. The arr_time is before the b_time. If the wait_time > Δtime , then no influence

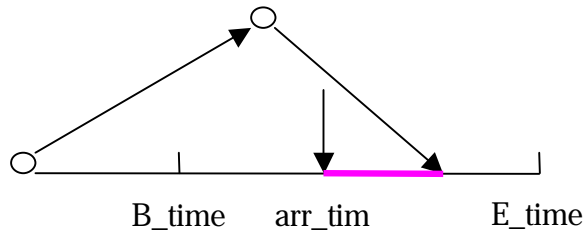
to this city, and the cities after this city will not be influenced. The Δ time will be 0. Refer to **Figure 19**. If the $\text{wait_time} < \Delta$ time, then part of the Δ time is cancelled by the wait_time . Therefore, the new Δ time will be smaller to influence the later cities. For this city, new arr_time will be the $\text{b_time} + \text{new } \Delta$ time, refer to **Figure 20**.

If Δ time < 0 :

1. If the arr_time is later than the b_time , then subtract the Δ time. If the Δ time is small, move the arr_time forward. Refer to **Figure 21**. If the Δ time is big, the arr_time will be before its b_time , then adjust the Δ time to be smaller. The new arr_time is the b_time . Refer to **Figure 22**.
2. If the arr_time is the b_time , then Δ time = 0, and make no changes.

— Delta time >

— Delta time < 0



Figure

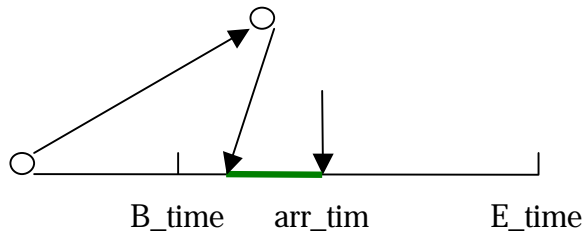


Figure 14

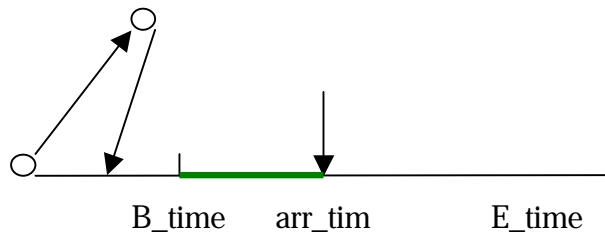


Figure 15

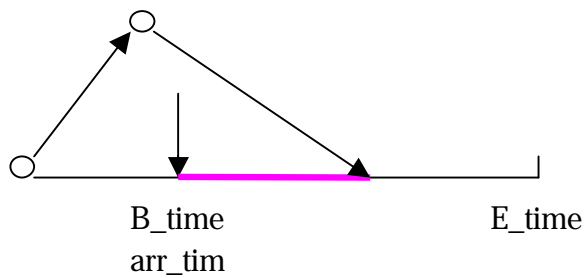


Figure 16

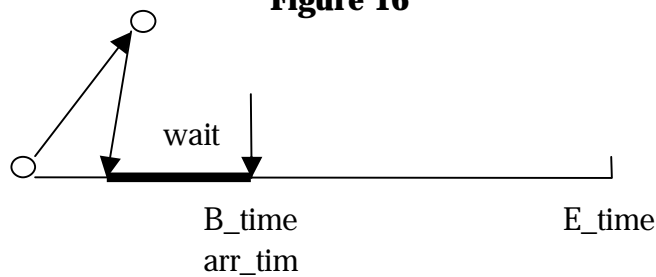


Figure 17

Delta time =

— Delta time < 0 — Delta time > 0
— New delta - - - → New

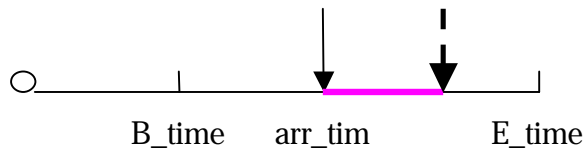


Figure 18

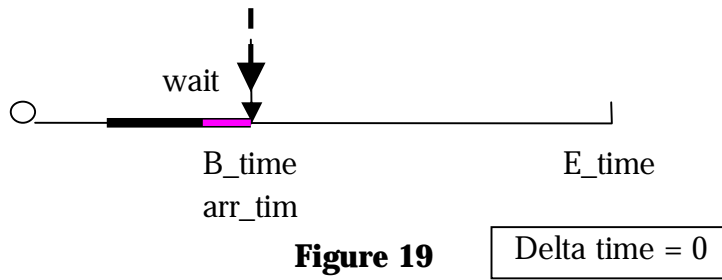


Figure 19

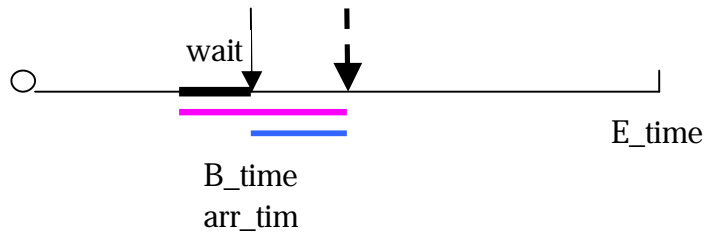


Figure 20

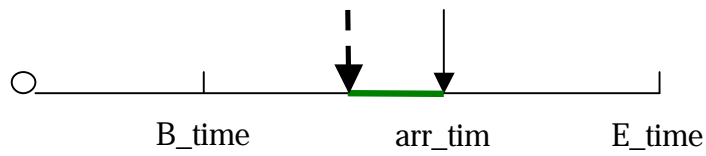


Figure 21

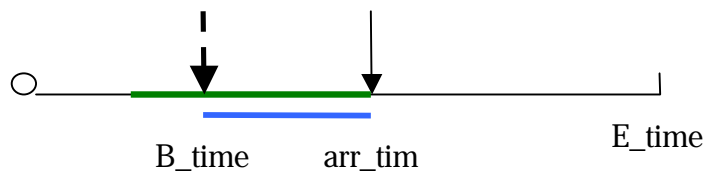


Figure 22

3.4.3 Structure of the ACS for Vehicle

3.4.3.1 Building Model

ACS_Vehicle colony searches for a feasible solution by maximizing the number of visited customers. ACS_Vehicle starts its computation using $v-1$ vehicle passed by global parameter, that is, one vehicle less than the smallest number of vehicles for which a feasible solution has been computed. During this search the colony produces unfeasible solutions in which some customers are not visited. In ACS_Vehicle, the solution computed since the beginning of the trial with the highest number of visited customers is stored in the variable $\psi^{ACS_Vehicle}$. A solution is better than $\psi^{ACS_Vehicle}$ only when the number of visited customers is increased. Therefore ACS_Vehicle is different from the traditional ACS applied to the TSP. In ACS_Vehicle the current best solution $\psi^{ACS_Vehicle}$ is the solution (usually unfeasible) with the highest number of visited customers, while in ACS applied to TSP the current best solution is the shortest one. In order to maximize the number of customers serviced, ACS_Vehicle use a set of variable IN_i . The entry IN_i stores the number of time customer i has not been inserted in a solution. IN is used by the constructive procedure `new_active_ant` for favoring the customers that are less frequently included in the solutions (discussed in 3.4.3.2). In ACS_Vehicles, at the end of each cycle, pheromone trails are globally updated with two different solutions: $\psi^{ACS_Vehicle}$, the unfeasible solution with the highest number of visited customers and ψ^k , the feasible solution with the lowest number of vehicles.

Here is the **pseudo code 3** for the Main ACS_Vehicle:

```

Procedure ACS_Vehicle
/* Initialization*/
Begin with v-1 vehicles, initialize pheromone and ants.
 $\psi^{ACS\_Vehicle} \leftarrow$  initial solution with v-1 vehicles produced with a nearest neighbor
    heuristic (may not be feasible, record the number of unvisited
    customers

/*Loop*/
For i < Max_iteration
    For each ant k
        /* Construct a solution  $\psi^k$  */
        new_active_ant (k, pheromone, customers, IN)
         $\forall$ Customer  $j \notin \phi^k : IN_j \leftarrow IN_j + 1$ 
    EndFor
    If exist k: #visited_customer ( $\psi^k$ ) > #visited_customer( $\psi^{ACS\_Vehicle}$ )
        then  $\psi^{ACS\_Vehicle} \leftarrow \psi^k$ 
            Reset  $IN_j$  to be 0, any j
    EndIf
    Do twice global pheromone update according to equation 2.3 for both
     $\psi^{ACS\_Vehicle}$  and  $\psi^{global}$  ( $L^+$  is calculated by both  $\psi^{ACS\_Vehicle}$  and  $\psi^{global}$ )
EndFor

```

Pseudo Code 3

3.4.3.2 Details of the ACS_Vehicle

In 3.4.3.1, there is a construction function “new_active_ant”. This function is the main process of the ant system. ACS_Vehicle and ACS_Time(discuss later) both use this function.

This constructive procedure is similar to the ACS constructive procedure designed for the TSP: each artificial ant starts from a randomly chosen copy of the depot and, at each step, moves to an unvisited node that does not violate time window constraints and vehicle capacities. The set of available nodes, in case the ant is not located in a

duplicated depot, also includes the unvisited duplicated depots. An ant at customer i chooses next customer based on equation (2.1) and (2.2). But the visibility η_{ij} is computed by taking into account the traveling time t_{ij} between nodes i and j , the time window $\{b_j, e_j\}$, the urgency the customer to be served and the number of times IN_j node j has not been inserted in a problem solution.

$$\text{delivery_time} \leftarrow \max(\text{current_time} + t_{ij}, b_j) \quad (3.12)$$

$$\text{delta_time} \leftarrow \text{delivery_time} - \text{current_time} \quad (3.13)$$

$$\text{distance} \leftarrow \text{delta_time} * (e_j - \text{current_time}) - IN_j \quad (3.14)$$

$$\eta_{ij} = 1/\text{distance}$$

In (3.12) and (3.13), delta_time is the time window from “now” to the next arriving time at customer j which takes the waiting time into account.

In (3.14), $(e_j - \text{current_time})$ calculate the urgency of the customer j to be served, if it is a customer whose time window will expire soon, then this customer has more priority.

The urgency mechanism also can reduce the attraction of depot or duplicated copies of depot. Because the e_j of depot is the work_end of a day, it is the latest time of working hour. Therefore, depot is usually selected when there is no candidate customers exist because of time and demand constraints. IN_j is a parameter to memorize how many times the customer has not been visited; it helps to favor more the customer who has not been visited in previous ants' work.

η_{ij} is the inverse of the distance, the shorter the distance, the more chance it will be selected. In all these equations, the mathematical calculations are based on numerical number of minutes.

Refer to **Pseudo Code 4**

```

/* new _active_ant: Constructive procedure for ant k used by ACS_Vehicle and
ACS_Time
/* Initialization*/
    put ant k in a randomly selected duplicated depot i
    current_time  $\leftarrow$  Work_Begin, load  $\leftarrow$  0

/* This is the step in which ank builds its tour. Tour is stored in  $\psi^k$ */

Loop
    /* Starting from node I compute the set  $N$  of feasible nodes (i.e. all the nodes j still to
    be visited and such that current_time and load are compatible with time windows
     $[b_j, e_j]$  and delivery quantity  $q_j$  of customer j)

    For any city in the feasible set  $N$ , Calculate its visibility  $\tau_{ij}$  as follows:

    delivery_time  $\leftarrow$  max(current_time+tij, bj)
    delta_time  $\leftarrow$  delivery_time – current_time
    distance  $\leftarrow$  delta_time * (ej-current_time) - INj /* product are in minutes format*/
     $\tau_{ij} = 1/\text{distance}$ 

    Choose probabilistically the next node j using  $\tau_{ij}$  in exploitation and exploration by
    equation (2.1) and (2.2)

    add the selected city to  $\psi^k$ 

    Current_time  $\leftarrow$  delivery_time
    load  $\leftarrow$  load+qj

    IF j is a depot of copy of depot
        THEN current_time  $\leftarrow$  work_begin, load  $\leftarrow$  0

    Do local pheromones update according to equation (2.4) and (2.5)
Loop END until no feasible candidate nodes exist

FOR unvisited nodes exist and this ACS does not fail
    FOR still has routes which have not been tried
        Do insertion according to equation (3.11)
        IF insertion success
            Break;
        END FOR
    IF insertion not success
        THEN this ACS fails and Return false;
    ENDIF
ENDFOR until all unvisited nodes have been inserted
Return true

```

3.5 Ant Colony to Minimize Transportation Time

ACS_Time is activated when fewer vehicles can not be found after a certain number of iteration. Its objective is to minimize the total transportation time using the feasible fewest vehicles generated by previous ACS_Vehicles. ACS_Time colony is a traditional ACS based colony whose goal is to compute a tour as short as possible. In ACS_Time m artificial ants are activated to construct problems solutions ψ^1, \dots, ψ^m . Each solution is build by calling the new_active_ant procedure explained in details in section 3.4.3.2. Although in this ACS, a final feasible solution is guaranteed, unfeasible solution will occur during several ants' work. Therefore, we still need insertion in ACS_Time. Once all the ants finish their work in a ACS_Time, we select the best ant with the shortest transportation time. It is compared to ψ^{global} , it is better than the global solution, then update the global solution and begin another ACS_Time until a stopping criterion is met . Refer to **pseudo code 5**.

```
/* ACS_Time */
/* parameter v is the smallest number of vehicles for which a feasible solution
has been computed */
/* Initialization */
  Initialize pheromone and data structures using v
/* Cycle */
  Repeat
    FOR each ant k
      /* construct a solution  $\psi^k$  */
      new_active_ant
    ENDFOR each
  /* update the best solution if it is improved */
  IF exist ant k ,  $\psi^k$  is feasible and time of k < global time
    THEN update the global best path
    Perform global pheromone update according equation (2.3)
  Until a stopping criterion is met
Pseudo code 5
```

3.6 Adaptation to Real Life Application

3.6.1 Limitation

The results generated by the multi – objective VRPTW are restricted by the vehicle capacity. e.g. if the total demand of 100 customer is 250 units, each vehicle has a capacity of 50 units, then at least 5 vehicles are needed. But under a normal width of time window, the nearest neighbor heuristics has already generated a solution with 5 vehicles. Therefore, the multi – objective ACS has no effect on minimizing the vehicle number.

3.6.2 Hong Kong Geographical Characteristic

This project tries to simulate a delivery system within HK area. However, the VRP problem is first introduced in the United States delivery system. US have a more wide land area and their customers are comparatively far away from each other. The vehicle speed in US is rather fast because most of its roads are highway. Hong Kong is a city with land area around 1000 km², it has a population around 6 million. Therefore, it's a very compact and clustered area; customers may be in a single building. The traffic situation is also different from US. In my project, I assume the average speed of vehicle is 50km/h, therefore the average delivery time in HK is within one hour, usually 20 to 40 minutes.

Besides that, the depot in HK is close to customers. Therefore, why not let vehicles have a chance to reload when its capacity is finished or nearly finished? This can solve the restriction problem as stated before. In real life, it is more economical and reasonable.

3.6.3 Modification of Building model

In my modification, I allow each vehicle has a chance to reload at most once. But it is not arbitrary, the driver can select to reload or he can give up this chance. It depends on the cooperative work of the whole system.

To achieve this, I made another 4 “shadow” of the original depot and the copies of the depots. Then totally there are 8 nodes with the position of the depots. Refer to **figure 23**. The red node is the original depot; the yellow nodes are copies of the depots and the grey one are shadows of the red and yellow nodes.

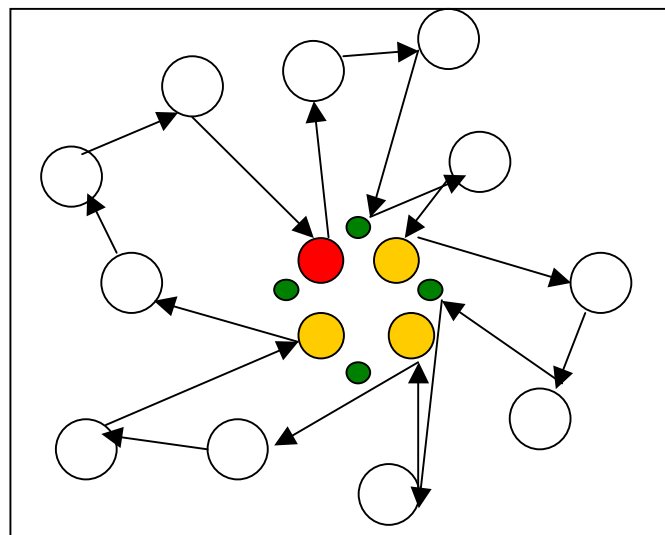


Figure 23

In **figure 23**, there are still 4 vehicles used compare to **figure 6**. 2 vehicles choose to reload at the depot, represent by green dots in the graph, and another 2 vehicles choose not to reload. Therefore, the other 2 green dots have not been visited in this graph.

In **figure 6**, an artificial ant are required to visit all the nodes in the graph including the copies of the depots, but here, the green dots can remain vacant because the driver can choose to reload or not. Other nodes are still required to be visited by the ants.

3.6.4 Change in the visibility function

As indicated in the section 3.4.3.2, each ant will choose the next cities based on equation (2.1) and (2.2), among this, the visibility η_{ij} is calculated by equation (3.12),(3.13) and (3.14).

Because now we add another sets of shadows, and these shadows are supposed not to influence the visibility of traditional customers. Therefore, we must reduce the visibility of these shadows. Because the shadow has the same time window as the depot,[work_begin, work_end], but they have higher priority than the copy of depot. In equation (3.14), we count in the urgency ($e_j - \text{current_time}$), to increase the priority of the shadows, we calculate the urgency of the shadow by $3 * \text{delta_time}$ (3.13). Now the new equation for visibility of shadow is (3.15). The advantage is that, the visibility of the shadow is lower than traditional customer and higher than copy of depot. Therefore, it provides chances for drivers to return depot for reloading. Of course, the driver can give up the chance; it depends on the whole performance of the ACS.

$$\begin{aligned}
 \text{delivery_time} &\leftarrow \max(\text{current_time} + t_{ij}, b_j) \\
 \text{delta_time} &\leftarrow \text{delivery_time} - \text{current_time} \\
 \text{distance} &\leftarrow \text{delta_time} * \{ 3 * \text{delta_time} \} \\
 \eta_{ij} &= 1/\text{distance}
 \end{aligned}
 \tag{3.15}$$

Originally, when the next node is depot or copies of the depot:
 $\text{currnet_time} \leftarrow \text{Work_begin}$
 $\text{load} \leftarrow 0$

But if the node is a shadow now:
 $\text{current_time} \leftarrow \text{current_time} + \text{traveling time} + \text{service time}$
 $\text{load} \leftarrow 0$

It is exactly the situation in real life, when a vehicle return to reload, it will waste a certain period of time, but the vehicle capacity has been refreshed.

Chapter 4

Experiment Result

4.1 Data Set

I choose the benchmark data set introduced by [Solomon 1987]. His data has 2 main types: clustered distributed and randomly distributed. His data has time window longer than 1 day, therefore, I only choose those within 1 day delivery.

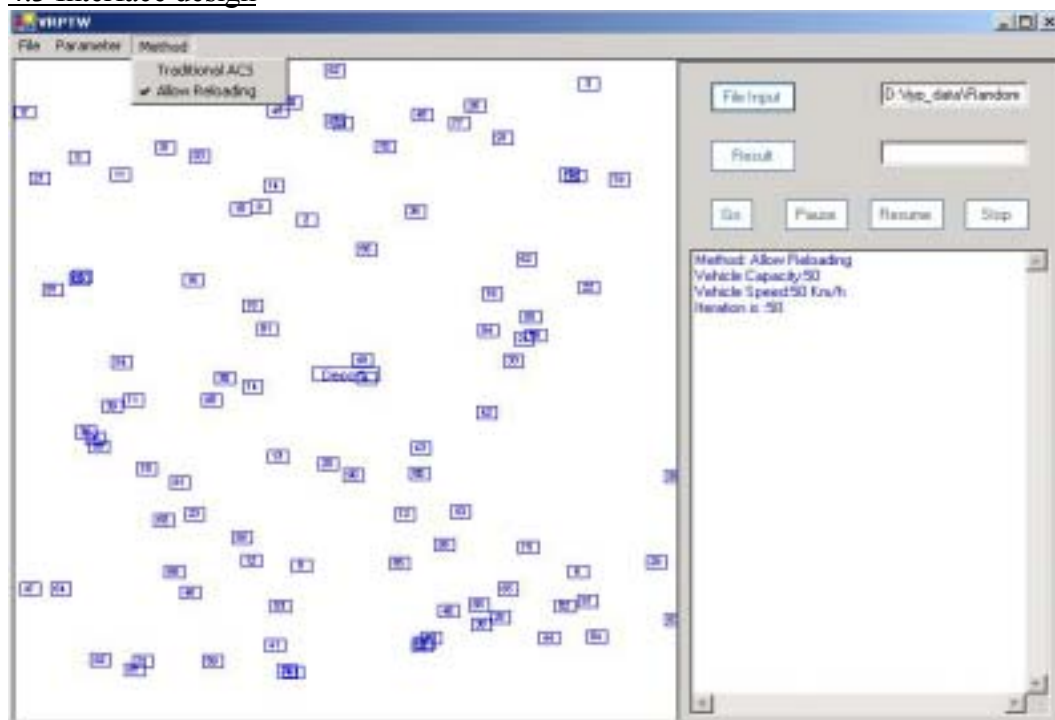
4.2 Parameter Setting

Parameters have subtly influence on the final results.

- Vehicle capacity – it directly influence the number of fleet used. If the vehicle capacity is large, then the whole problem depends on the width of time window. The difference between traditional methods and adapted once is small. If the vehicle capacity is small, then the problem is restricted by its capacity. Then the adapted method has great advantage over traditional method.
- Vehicle speed – it will influence the time constraints. If the vehicle speed is too slow, then the time window is indirectly shortened which will increase the selection pressure. If the vehicle speed is too fast, it is not practical in daily life and the solution is too optimized to be true.
- Exploitation rate q_0 – in the selection phase of the next customer, we use equation (2.2) to increase the exploration for new solutions. Normally we set q_0 to be 0.9. Therefore, it has 10% space to explore new solutions. We can increase this percentage, say 30%, then it has more variance and it act more like a random search.

- Pheromones update $\rho - \rho$ is used in the local pheromone update and also global pheromone update. In local pheromone update, its task is to reduce the attraction of most visited routes to prevent convergence. In global pheromone update, its task is to reinforce the best path. Therefore, we select a mediate value, 0.1. Refer to equation (2.3),(2.4) and(2.5)
- Ant number – we set the ant number as equal to the customer number. In the testing data, 100 customer Vs 100 artificial ant. Because in each ACS, 100 ants has to finish the whole route, and the normal iteration is 50, therefore, $50 * 100 = 5000$ iteration. Too many ants will increase the calculation pressure while fewer ants can't reflect the cooperative work of the colony.

4.3 Interface design




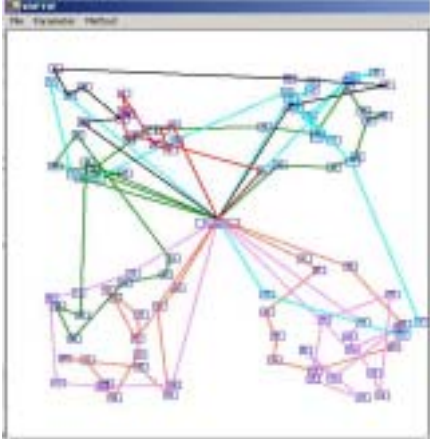
Figure

In figure 24, there are 3 components. The panel which displays the position of the customers, a result list which show the parameter setting and later show the whole result list, and menu buttons allow you to input file and start the thread.

Each color of lines represents one vehicle.

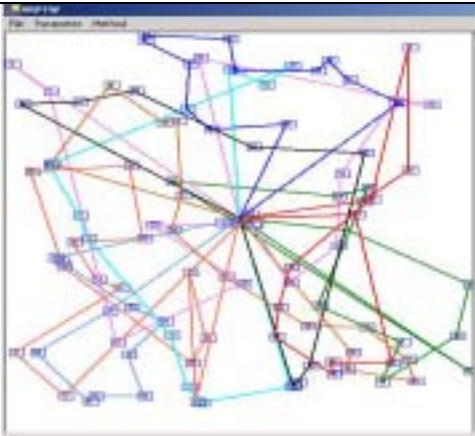
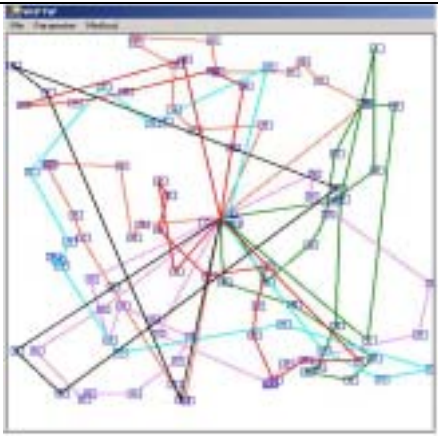
4.4 Testing Result

4.4.1 Clustered Data

Traditional Methods	Allow Reloading
	
9 vehicles, total transportation time is 74h 36 mins, total waiting time is 25h 33 mins	6 vehicles, total transportation time is 73h40mins, total waiting time is 6h23mins

Figure

4.4.2 Random Data

Traditional Methods	Allow Reloading
	

9 vehicles, total transportation time is 74h18mins, total waiting time is 19h57mins	6 vehicles, total transportation time is 72h19mins, total waiting time is 7h16mins
---	--

Figure

4.4.3 Comparison of parameter changes

The default parameter is:

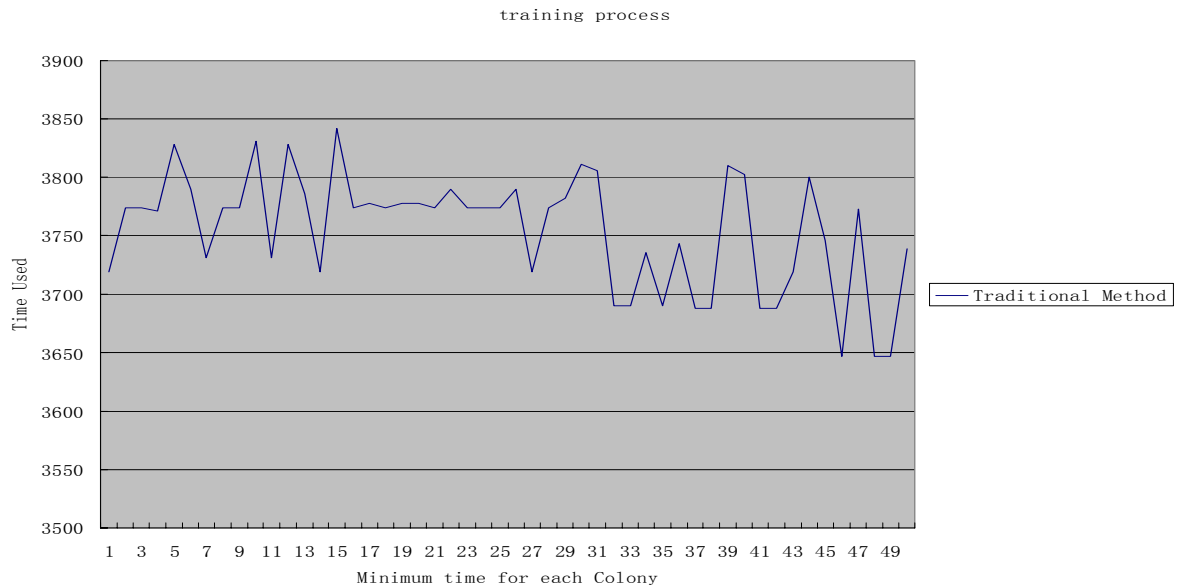
Capacity = 50 units; Speed = 50Km/h; 100 customers randomly distributed;

$q_0=0.1$; $\rho =0.1$;

	Traditional Methods		Allow Reloading	
Capacity=50	9 vehicles	74h 36mins	6 vehicles	73h40mins
Capacity=100	6 vehicles	72h4mins	6 vehicles	74h30mins
Speed = 40km/h	9 vehicles	76h15mins	8 vehicles	76h30mins
Speed = 60km/h	9 vehicles	67h54mins	5 vehicles	56h26mins
$\rho =0.3, q_0=0.8$	9 vehicles	72h 18mins	6 vehicles	73h 40mins

Figure

4.4.4 Performance of the ACS in the training process



Figure

4.4.5 Analysis

- In the clustered data and the random data, the total transportation time of traditional methods are near those of the adapted methods. But the adapted methods use 30% fewer vehicles than the traditional methods. Besides, the waiting time is largely reduced which increase the work efficiency.
- The interface shows a 2-D Euclidian space which only includes the physical position of the customers. Actually, the customer is restricted by time window; therefore, there are some long distance connections on the graph. Most customers who live near are visited by the same car, but special situation occurs because of the time window.
- The result in 4.4.3 shows the influence of the parameters. Obviously, the vehicle number of the traditional method is restricted by the vehicle capacity. No matter how fast the vehicles are, they still need 9 vehicles unless the capacity is enlarged. When the capacity reaches 100, then time window is the major constraints. Therefore, there is no difference between 2 methods in the vehicle number if the capacity is 100 because in this case, no driver will select reload at the depot. The increase of exploitation parameters help the traditional method to find a shorter path based on the same other parameters, but there is no help in the adapted methods. Each time run the program, the result maybe slightly different or sometimes same as previous. This is because ACS contains randomness. Each result in figure 27 is based on average time of 5 results.
- Figure 28 shows the training process of the ACS in traditional method. I triggered the best transportation time after each ACS_Time finishes. Therefore, there are 50 values total. From the graph, the value does not converge, it has many variances.

But the trend of the value is decreasing no matter the maximum value or the minimum value. The ACS is a good guided search based on pheromone, it is also the evaporation of pheromone which prevent the convergence. The graph is generated when q_0 is 0.1 and $\rho = 0.1$.

Chapter 5

Conclusion

5.1 My Contribution

In my project, I start my project with Traveling Sales Problem, then to Vehicle Routing Problem and finally to the Vehicle Routing Problem with Time Windows. I successfully achieve the basic VRPTW and make application to real life problem.

In section (3.4.2.2), I try to modify the Solomon' method by considering capacity factors when select the candidate insertion customer. This can save the running time if the ACS doomed to fail, or it let the customer with large demand to be inserted first.

In section (3.4.3.3), I try to do insertion with consideration of different time constraint and its influence on the later city.

After achieving my project at the first stage, I find restriction of capacity in the result. Then I try to modify the program to apply in real life. Hong Kong is a compact area which allows reloading feasible. Therefore, based on the model of figure 6, I think out the "shadow" method (figure 23) which allow the driver choose reloading. Because the visibility of the shadow is between normal customers and the depot, the driver can either choose or not choose reloading. The final result is that: some drivers reload between route and some are not. This situation is reasonable in real life.

Ant Colony System is a newly appeared method. Therefore, there are limited source of this method exist. Moreover, the issues about its application in Vehicle Routing Problem are even fewer and not mature. Through the study of this project, I try to

investigate what is nearest neighbor heuristic, what is cheapest insertion, greedy insertion, how to achieve this methods in my project with many constraints.

5.2 Discussion About VRPTW and ACS

Vehicle Routing Problem is an old and difficult question. It has been extended to many other application. VRPTW is one of its extensions of time constraints. Because of its time constraints, it's rather difficult and not effective to solve the unfeasible situation that some customers are not visited. e.g. Although in section (3.4.2.1) and (3.4.2.2), the insertion method is introduced, there are still some situation unsatisfactory. If an unvisited customer has a demand of 5 unites. There are 4 routes exist and each of them has a space left of 2 or 3 unites. The total space of the whole ACS is larger than 5, but they are split, and because of the time constraints, any swapping will be not economical and have to consider the influence of time. Therefore, a feasible solution may not be generated.

ACS has contribution to find the shortest path, but it seems not have many affect on the time constraints. ACS performs well in the TSP because there is no case for unfeasible solution. But in VRPTW, an insertion of customers violates the system of the ACS because the insertion is not based on the pheromone and the visibility of the customers. Therefore, during the ACS_Vehicles, it is not effective and difficult to find a feasible solution when $(v-1)$ vehicles are provided.

However, ACS is a simpler method compare to Genetic Algorithm and Neural Network. It uses simple agent (artificial ant) to achieve the goal based on cooperative work. The update and selection function of ACS are linear formula, therefore, the computation time is not so long. A common PC can get a reasonable result within a limited time.

5.3 Future Development

1. In my project, there is only one depot. However, in real life, there may be more than one depot. Therefore, the multi- depots solution can be developed.
2. In real life situation, the vehicle number may be fixed. It is doomed that some customers may not be visited within their time windows. We can classify the customers into different categories. We may sacrifice some low priority customers if it is needed.
3. Extend delivery time within 1 week. In my project, the delivery time is within 1 day. For some special products, the delivery time can be extended within 1 week or even 1 month.

References:

1. M.Dorigo et al.(Eds.): ANTS 2002, LNCS 2463, pp.53-64,2002 © Springer-Verlag Berlin Heidelberg 2002
2. M.Solomon, Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints, *Operations Research* 35,1987,254-265
3. J.Y.Povin, S.Bengio, The Vehicle Routing Problem with Time Windows – Part II: Genetic Search, *INFORMS Journal of Computing* 8, 1996,165-172
4. Ant Colony for the traveling salesman problem TR/IRIDIA/1996-3 Universite libre de Bruxelles, Belgium
5. Dorigo, M., Maniezzo, V. and Colorni, A., 1996, The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 26, 29–41.
6. MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. Technical Report IDSIA -06-99, Lugano, Switzerland, 1999
7. P. Kilby, P. Prosser, P. Shaw, Guided Local Search for the Vehicle Routing Problems With Time Windows, in *Meta-heuristics: Advances and Trends in Local Search for Optimization*, S.Voss, S. Martello, I.H. Osman and C.Roucairol (eds.), Kluwer Academic Publishers, Boston, 1999, 473-486.
8. B. Bullnheimer, R. F. Hartl, C. Strauss, An Improved Ant System Algorithm for the Vehicle Routing Problem, Technical Report POM-10/97, Institute of Management Science, University of Vienna, Austria, 1997. Accepted for publication in *Annals of Operations Research*.
9. R. Cordone, R. Wolfler-Calvo, A Heuristic for the Vehicle Routing Problem with Time Windows. To appear in *Journal of Heuristics*.
10. M. Dorigo, V. Maniezzo, A. Colorni, The Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics–Part B* 26, 1996, 29-41.
11. M. Dorigo, G. Di Caro, L. M. Gambardella, Ant Algorithms for Discrete Optimization, Technical Report IRIDIA/98-10, Université Libre de Bruxelles, Belgium, 1998. Accepted for publication in *Artificial Life*.
12. É. D. Taillard, FANT: Fast Ant System, Technical Report IDSIA-46-98, IDSIA, Lugano, Switzerland, 1998.
13. *Swarm Intelligence: From natural to artificial systems*/Eric Bonabeau, Marco Dorigo, Guy Theraulaz. New York: Oxford: Oxford University Press,1999
14. Bullnheimer, B.,Hartl,R.and Strauss, C: *Applying the Ant System to the Vehicle Routing Problem. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization* Kluwer:Boston(1997)
15. An Experimental Study of a Simple Ant Colony System for the Vehicle Routing Problem with Time Windows. Ismail Ellabib, Otman A.Basir, Paul Calamai

Appendix 1: Sample Result of Traditional Method

index/demand/arrive time/time window/x position/y position

Depot		8:30	500	500		
77	9	11:30	[11;30,14:0]	666	102	
27	8	12:1	[11;30,14:0]	490	102	
60	1	12:54	[12;0,15:0]	843	189	
97	7	13:19	[12;0,15:45]	733	126	
26	4	13:36	[12;0,16:0]	691	75	
48	3	13:55	[13;30,16:0]	610	89	
70	2	14:14	[13;0,18:0]	555	140	
51	5	14:34	[13;0,16:0]	478	98	
56	8	15:9	[15;0,17:30]	526	309	
83	1	15:54	[15;0,17:0]	275	156	
20	1	16:23	[16;0,18:0]	412	69	
49	1	17:24	[16;0,18:0]	520	488	

Depot		8:30	500	500		
13	1	9:30	[9;30,11:30]	392	647	
85	4	10:10	[10;0,13:0]	577	821	
95	1	10:28	[10;0,13:0]	645	790	
72	4	10:47	[10;30,12:30]	584	740	
8	2	11:18	[10;30,13:30]	429	825	
53	6	11:36	[10;20,13:0]	397	891	
93	9	12:30	[12;30,14:40]	624	943	
76	7	13:48	[13;30,16:0]	717	381	
4	6	14:30	[14;30,17:30]	526	518	
80	9	15:1	[15;0,18:0]	606	675	

Depot		8:30	500	500		
75	1	9:30	[9;30,12:30]	771	795	
65	3	9:48	[9;0,12:0]	740	863	
25	8	10:3	[9;30,12:30]	729	909	
84	8	10:30	[8;30,11:0]	875	940	
57	6	10:47	[10;0,14:0]	860	882	
63	1	11:13	[9;30,11:30]	993	912	
6	8	12:0	[12;0,18:30]	847	835	
44	5	12:23	[12;0,15:0]	803	942	
88	1	12:47	[12;0,14:0]	697	888	
38	9	13:0	[10;0,14:0]	701	921	

Depot		8:30	500	500		
68	1	9:25	[9;0,15:30]	220	750	
64	9	9:57	[9;0,12:0]	65	862	
31	5	10:30	[10;30,13:30]	189	980	

67	1	10:42	[10;0,14:30]	177	994
89	8	11:6	[10;0,14:15]	294	980
46	4	11:30	[10;30,14:0]	260	868
59	8	11:54	[11;30,13:30]	339	778
23	3	12:13	[10;0,13:0]	267	740
66	6	12:34	[12;30,15:30]	237	834
78	1	13:4	[13;0,16:30]	194	666
58	1	13:23	[10;30,13:30]	122	631
36	3	13:36	[11;0,14:0]	102	607

Depot	8:30	500	500		
35	6	9:12	[9;0,11:0]	600	246
2	4	9:41	[9;30,13:30]	436	260
14	1	9:59	[9;0,12:0]	386	205
15	7	10:32	[10;0,12:0]	354	401
81	7	10:47	[10;30,13:30]	377	438
74	3	11:8	[10;0,12:0]	355	532
96	8	12:0	[12;0,16:0]	116	617
39	5	12:30	[12;30,15:45]	142	564
98	3	13:0	[13;0,16:0]	293	554
30	6	14:0	[14;0,18:0]	312	519

Depot	8:30	500	500		
24	1	9:12	[9;0,12:0]	765	454
22	3	9:37	[9;0,12:30]	863	370
3	1	10:26	[9;0,13:0]	862	38
86	9	11:26	[11;0,14:0]	782	448
55	2	12:0	[12;0,14:30]	775	418
94	2	12:18	[12;0,15:0]	710	441
43	5	12:53	[10;30,14:40]	609	632
28	6	13:20	[10;0,15:0]	469	658
90	5	13:34	[13;30,15:45]	506	676
10	2	14:4	[12;0,15:0]	669	736
40	5	14:33	[14;30,16:45]	651	898
1	6	14:50	[14;0,19:0]	615	950
82	3	15:26	[13;30,16:0]	827	890

Depot	8:30	500	500		
34	6	9:40	[8;30,11:0]	406	996
47	2	12:0	[12;0,15:45]	19	862
92	1	14:0	[14;0,16:30]	125	978
61	6	15:0	[15;0,18:0]	416	998
41	2	15:30	[15;30,17:30]	388	955
12	3	15:57	[15;30,18:30]	353	817
71	7	16:45	[15;30,18:0]	175	554
54	9	17:2	[15;0,17:30]	156	492

69	4	17:29	[14;30,17:30]	93	356
21	3	17:59	[15;30,18:0]	33	192
11	3	18:23	[16;30,19:0]	155	186

Depot	8:30	500	500		
52	1	9:38	[9;0,11:30]	479	17
45	1	10:1	[10;0,13:0]	392	83
7	7	10:24	[10;0,14:0]	296	22
73	1	10:35	[10;0,13:0]	296	6
5	1	11:15	[10;30,13:30]	93	158
17	2	11:38	[10;0,14:30]	11	85
87	9	12:23	[12;0,15:0]	52	375
32	1	12:38	[12;0,16:0]	96	352
16	9	13:8	[12;30,15:30]	264	358
18	1	13:34	[10;30,15:30]	336	242
9	4	13:47	[11;30,14:30]	366	238
33	1	14:52	[13;30,17:0]	749	490
100	9	15:7	[15;0,18:0]	733	532
79	2	16:2	[15;0,18:0]	908	195
19	1	16:21	[14;30,17:30]	833	186

Depot	8:30	500	500		
37	1	12:0	[12;0,15:0]	222	142
91	3	14:0	[14;0,17:30]	245	689
42	4	15:7	[14;0,16:30]	709	575
62	8	15:47	[15;30,17:30]	768	325
99	5	16:47	[15;0,18:0]	993	679
29	9	17:14	[14;0,18:0]	965	819
50	7	18:9	[16;30,18:30]	610	954

The total time used is:3day0:38

The total waiting time is:0day11:19

Appendix 2: Sample Result If Allow Reloading

index/demand/arrive time/time window/x position/y position

Deport	8:30	500	500		
34	6	9:40	[8;30,11:0]	406	996
89	8	10:3	[10;0,14:15]	294	980
53	6	10:29	[10;20,13:0]	397	891
8	2	10:47	[10;30,13:30]	429	825
59	8	11:30	[11;30,13:30]	339	778
23	3	11:49	[10;0,13:0]	267	740
68	1	12:4	[9;0,15:30]	220	750
46	4	12:28	[10;30,14:0]	260	868
66	6	12:42	[12;30,15:30]	237	834
78	1	13:12	[13;0,16:30]	194	666
39	5	13:35	[12;30,15:45]	142	564
Return	14:28	500	500		
4	6	14:41	[14;30,17:30]	526	518
80	9	15:12	[15;0,18:0]	606	675
90	5	15:34	[13;30,15:45]	506	676
12	3	16:8	[15;30,18:30]	353	817
41	2	16:35	[15;30,17:30]	388	955
61	6	16:51	[15;0,18:0]	416	998
50	7	17:24	[16;30,18:30]	610	954
1	6	17:34	[14;0,19:0]	615	950

Deport	8:30	500	500		
24	1	9:12	[9;0,12:0]	765	454
22	3	9:37	[9;0,12:30]	863	370
3	1	10:26	[9;0,13:0]	862	38
86	9	11:26	[11;0,14:0]	782	448
55	2	12:0	[12;0,14:30]	775	418
94	2	12:18	[12;0,15:0]	710	441
43	5	12:53	[10;30,14:40]	609	632
10	2	13:17	[12;0,15:0]	669	736
38	9	13:49	[10;0,14:0]	701	921
93	9	14:8	[12;30,14:40]	624	943
40	5	14:30	[14;30,16:45]	651	898

Deport	8:30	500	500		
35	6	9:12	[9;0,11:0]	600	246
2	4	9:41	[9;30,13:30]	436	260
14	1	9:59	[9;0,12:0]	386	205
45	1	10:23	[10;0,13:0]	392	83
7	7	10:46	[10;0,14:0]	296	22
73	1	10:57	[10;0,13:0]	296	6

27	8	11:32	[11;30,14:0]	490	102
77	9	12:3	[11;30,14:0]	666	102
26	4	12:17	[12;0,16:0]	691	75
97	7	12:34	[12;0,15:45]	733	126
60	1	12:59	[12;0,15:0]	843	189
33	1	13:46	[13;30,17:0]	749	490
Return		14:25	500 500		
28	6	14:54	[10;0,15:0]	469	658
30	6	15:29	[14;0,18:0]	312	519
98	3	15:43	[13;0,16:0]	293	554
71	7	16:7	[15;30,18:0]	175	554
54	9	16:24	[15;0,17:30]	156	492
69	4	16:51	[14;30,17:30]	93	356
21	3	17:21	[15;30,18:0]	33	192
11	3	17:45	[16;30,19:0]	155	186

Deport		8:30	500 500		
25	8	9:36	[9;30,12:30]	729	909
65	3	9:51	[9;0,12:0]	740	863
75	1	10:9	[9;30,12:30]	771	795
57	6	10:33	[10;0,14:0]	860	882
84	8	10:50	[8;30,11:0]	875	940
63	1	11:14	[9;30,11:30]	993	912
6	8	12:0	[12;0,18:30]	847	835
44	5	12:23	[12;0,15:0]	803	942
88	1	12:47	[12;0,14:0]	697	888
82	3	13:30	[13;30,16:0]	827	890
42	4	14:20	[14;0,16:30]	709	575
Return		14:56	500 500		
56	8	15:29	[15;0,17:30]	526	309
100	9	16:15	[15;0,18:0]	733	532
62	8	16:50	[15;30,17:30]	768	325
19	1	17:18	[14;30,17:30]	833	186
79	2	17:37	[15;0,18:0]	908	195

Deport		8:30	500 500		
13	1	9:30	[9;30,11:30]	392	647
74	3	10:0	[10;0,12:0]	355	532
15	7	10:25	[10;0,12:0]	354	401
81	7	10:40	[10;30,13:30]	377	438
18	1	11:14	[10;30,15:30]	336	242
9	4	11:30	[11;30,14:30]	366	238
37	1	12:0	[12;0,15:0]	222	142
5	1	12:25	[10;30,13:30]	93	158
17	2	12:48	[10;0,14:30]	11	85
87	9	13:33	[12;0,15:0]	52	375

32	1	13:48	[12;0,16:0]	96	352
16	9	14:18	[12;30,15:30]	264	358
83	1	15:0	[15;0,17:0]	275	156
70	2	15:43	[13;0,18:0]	555	140
20	1	16:12	[16;0,18:0]	412	69
Return		17:14	500	500	
49	1	17:26	[16;0,18:0]	520	488

Deport		8:30	500	500	
52	1	9:38	[9;0,11:30]	479	17
36	3	11:12	[11;0,14:0]	102	607
58	1	11:25	[10;30,13:30]	122	631
96	8	12:0	[12;0,16:0]	116	617
47	2	12:41	[12;0,15:45]	19	862
31	5	13:15	[10;30,13:30]	189	980
67	1	13:27	[10;0,14:30]	177	994
92	1	14:0	[14;0,16:30]	125	978
91	3	14:47	[14;0,17:30]	245	689
29	9	16:24	[14;0,18:0]	965	819
99	5	16:51	[15;0,18:0]	993	679

Deport		8:30	500	500	
64	9	9:47	[9;0,12:0]	65	862
85	4	10:58	[10;0,13:0]	577	821
72	4	11:17	[10;30,12:30]	584	740
95	1	11:36	[10;0,13:0]	645	790
51	5	13:11	[13;0,16:0]	478	98
48	3	13:36	[13;30,16:0]	610	89
76	7	14:23	[13;30,16:0]	717	381

The total time used is:2day15:1

Appendix 3: NP-Complete Problem

There are many tasks for which we know fast (polynomial) algorithms. There are also some problems that are not possible to be solved algorithmically. For some problems was proved that they are not solvable in polynomial time.

But there are many important tasks, for which it is very difficult to find a solution, but once we have it, it is easy to check the solution. This fact led to **NP-complete** problems. NP stands for nondeterministic polynomial and it means that it is possible to "guess" the solution (by some nondeterministic algorithm) and then check it, both in polynomial time. If we had a machine that can guess, we would be able to find a solution in some reasonable time.

Studying of NP-complete problems is for simplicity restricted to the problems, where the answer can be yes or no. Because there are tasks with complicated outputs, a class of problems called **NP-hard** problems has been introduced. This class is not as limited as class of NP-complete problems.

For **NP-problems** is characteristic that some simple algorithm to find a solution is obvious at a first sight - just trying all possible solutions. But this algorithm is very slow (usually $O(2^n)$) and even for a bit bigger instances of the problems it is not usable at all.

Today nobody knows if some faster exact algorithm exists. Proving or disproving this remains as a big task for new researchers). Today many people think, that such an

algorithm does not exist and so they are looking for some alternative methods - example of these methods are genetic algorithms.

Examples of the NP problems are satisfiability problem or knapsack problem.

Examples of NP-hard problems include the Hamiltonian cycle and traveling salesman problems.

Traveling salesman problems: a salesman has to visit all of his customers once, traveling the least distance possible.
