

Routing and Delivery Planning: Algorithms and System Implementation

WONG Chi Fat

黃志發

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Philosophy

in

Systems Engineering and Engineering Management

© The Chinese University of Hong Kong

April 2002

The Chinese University of Hong Kong holds the copyright of this thesis. Any person(s) intending to use a part or whole of the materials in the thesis in a proposed publication must seek copyright release from the Dean of the Graduate School.



Abstract

In this thesis, we focus on a map-based decision-support system dealing with routing and delivery planning in Hong Kong. The system provides a digital map-based interface, data-management and algorithms for delivery planning.

Delivery planning is a difficult problem in Hong Kong. High variation of the traffic conditions makes actual vehicle-travel-time hard to predict. Because of this, in this thesis, we propose one time-varying and one stochastic model to handle delivery planning in Hong Kong. The first one is the *Time-varying Constrained Shortest Path* (TCSP) and the second one is the *Vehicle Routing Problem with Time Windows and Stochastic Travel Time* (VRPTWST).

TCSP differs from traditional shortest path in that the transit duration of an arc in the network is a function of time. This adjustment can reflect the traffic condition at particular point of time and take this factor into consideration in evaluating the shortest path. One exact and one heuristic algorithm are proposed to solve the shortest path problem in a 2-level network.

VRPTWST is another model considered in the thesis. It is similar to the deterministic *Vehicle Routing Problem with Time Window* (VRPTW). The only difference is that the travel time between two nodes is a stochastic variable in VRPTWST. A two-stage stochastic recourse model and an exact branch-and-cut algorithm are proposed to solve the problem. Furthermore, some side-heuristics are suggested to improve the efficiency of the algorithm.

At the end of the thesis, some implementation issues of our system are discussed. A study is made on some commercial computerized vehicle routing software in the market.

摘要

本篇論文集中研究一套決策支援系統 (decision-support system) ，此系統專門處理一些與香港有關的路徑找尋及運送計劃問題 (routing & delivery planning problem) ，此系統提供電子地圖介面、資料庫管理及一系列對運送計劃問題的計算解決方法。

運送計劃問題於香港是一個難以解決的問題。高度變化的路面交通情況令到真正的行車時間難以推算，因此在本文中，我們將提出一隨時間變化 (time-varying) 及一隨機性 (stochastic) 的模型用以處理與香港有關的運送計劃問題。它們分別是：一) 隨時間變化及有限制性的最短路徑問題 (Time-varying Constrained Shortest Path Problem – TCSP) 及二) 有時限性、隨機性行車時間的車輛路徑問題 (Vehicle Routing Problem with Time Windows and Stochastic Travel Times – VRPTWST) 。

隨時間變化及有限制性的最短路徑問題 (TCSP) 與一般最短路徑問題是不相同的。其分別於在一個網絡裏，前者通過兩節點之間所需的時間 (transit duration) 是一個與時間相關的函數 (a function of time)，而後者所需的時間卻是一個實數 (a fixed value)。這個改動能夠將當時的路面交通情況從函數中反映出來，從而將這因素考慮在計算最短路徑當中。我們將提出一準確的算法 (an exact algorithm) 及一啟發性算法 (a heuristic algorithm) 去計算在一個兩層網絡裏的隨時間變化及有限制性的最短路徑問題。

有時限性、隨機性行車時間的車輛路徑問題 (VRPTWST) 是另一個在本文中研究的問題。在這問題中，兩節點之間的行車時間是一個隨機變

數 (stochastic variable) 而非一個實數。一個兩階段隨機求助模型 (two-stage stochastic recourse model) 及一個準確的支切算法 (branch-and-cut algorithm) 將被提出去解決這問題。再者，我們更進一步建議一些輔助啓發算法 (side heuristics) 去改善原有算法的效率。

在本文尾部，我們將討論該系統的開發過程以及簡單探討一些在市面發售的車輛路徑計劃的軟件。

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Janny M.Y. Leung, not only for her valuable advice, but also for her patient teaching, encouragement and support. She offers me a lot of opportunities to build up myself and makes me know what is the spirit of knowledge. Besides, she also understands my strength, weakness and personality, so instructs me in a suitable and correct way. Her enlightenment is important and useful for my future.

At the same time, I would like to thank my department, Systems Engineering and Engineering Management, offering me an excellent atmosphere to study. The faculty members there are tolerant and teach me a lot of worthy knowledge. Also, a special thank is given to Professor C.H. Cheng who provides me a fantastic topic of my research and final year project.

Moreover, the development of the decision-support system is a joint work with another M. Phil. student, Mr. Paul T.W. Fung, and I would like to express my gratitude for his contribution of building up the database connection, digital maps, and other technical issues in the project.

At last, I would like to thank my parents providing me an excellent environment to live, and to learn. Their deep efforts make me not necessary to worry about my living so that I can straightly concentrate on my work. Also, my hearty thank is to my best friends including Dr. Ada Ng, Paul Fung, Angie Lim and Boris Li. They bring me lot of supports when I am in adversity, and lot of joys in my life.

Contents

List of Tables	ix
List of Figures	x
1. Introduction	1
1.1 Motivation	1
1.2 Literature Review	3
1.2.1 Shortest Path Problem	4
1.2.2 Vehicle Routing Problem with Time Windows	6
1.3 Thesis Outline	9
2. Time-varying Shortest Path with Constraints in a 2-level Network	11
2.1 Introduction	11
2.2 Problem Formulation of TCSP	12
2.3 Arbitrary Waiting Time	13
2.4 TCSP in a 2-level Network	16
2.4.1 Problem Formulation of TCSP in a 2-level Network	17
2.5 Algorithms Solving TCSP in a 2-level Network	20
2.5.1 Exact Algorithm	21
2.5.2 Heuristic Algorithm	23
2.6 Concluding Remarks	30
3. Vehicle Routing Problem with Time Windows and Stochastic Travel Times	32
3.1 Introduction	32
3.2 Problem Formulation	34

3.3 General Branch-and-cut Algorithm	42
3.4 Modified Branch-and-cut Algorithm	44
3.4.1 Prefixing	45
3.4.2 Directed Partial Path Inequalities	47
3.4.3 Exponential Smoothing	50
3.4.4 Fast Fathoming	54
3.4.5 Modified Branch-and-cut algorithm	56
3.5 Computational Analysis	57
3.5.1 Performance of Prefixing, Direct Partial Path Inequalities and Exponential Smoothing	57
3.5.2 Performance of Fast Fathoming	63
3.5.3 Summary of Computational Analysis	67
3.6 Concluding Remarks	67
 4. System Features and Implementation	 69
4.1 Introduction	69
4.2 System Features	70
4.2.1 Map-based Interface and Network Model	70
4.2.2 Database Management and Query	73
4.3 Decision Support Tools	75
4.3.1 Route Finding	76
4.3.2 Delivery Planning	77
4.4 System Implementation	80
4.5 Further Development	82
 5. Vehicle Routing Software Survey	
5.1 Introduction	83
5.2 Essential Features in CVRS Nowadays	84
5.2.1 Common Features	84
5.2.2 Advanced Features	90
5.3 Concluding Remarks	94

6. Summary & Future Work 97

Appendix A 101

Appendix B 104

Bibliography 107

List of Tables

3.1	Summary of the parameters of the sample problem when $n = 6$, $k = 2$ and $ \Xi = 3$	59
3.2	Summary of the average values of statistics of the sample problem when $n = 6$, $k = 2$ and $ \Xi = 3$	60
3.3	Summary of the difference between the original algorithm and enhanced algorithms with different techniques added when $n = 6$, $k = 2$ and $ \Xi = 3$	60
3.4	Summary of the average values of statistics of the sample problem when $n = 10$, $k = 2$ and $ \Xi = 3$	62
3.5	Summary of the difference between the original algorithm and enhanced algorithms with different techniques added when $n = 10$, $k = 2$ and $ \Xi = 3$	62
3.6	Summary of the average values of statistics of the sample problem when $n = 10$, $k = 2$ and $ \Xi = 3$	64
3.7	Summary of the difference between the PDE algorithm and enhanced algorithms with different fast fathoming techniques added when $n = 10$, $k = 2$ and $ \Xi = 3$	64
3.8	Summary of the average values of statistics of the sample problem when $n = 15$, $k = 2$ and $ \Xi = 3$	66
3.9	Summary of the difference between the PDE algorithm and enhanced algorithm with different fast fathoming techniques added when $n = 15$, $k = 2$ and $ \Xi = 3$	66
5.1	Summary of the feature differences between CVRS of different eras	89
5.2	Features of CVRS products investigated	94

List of Figures

2.1	Layout of 2-layer network	18
3.1	Graphical Presentation of Algorithm 3.2	43
3.2	Solution paths for P^1 and P^2	48
4.1	Map-based interface of VANS	71
4.2	Interface for adding new customers	74
4.3	Interface for adding new orders	75
4.4	Time-varying shortest path at different time intervals	76
4.5	Delivery report	79
4.6	Animated route displaying function	79
4.7	Relationships among the software used in VANS	81
5.1	Interfaces of some CVRS software	85

Chapter 1

Introduction

1.1 Motivation

Transportation plays a very important role in the economy of many developed countries. From the statistics of Hong Kong Special Administrative Region, (HKSAR) China in 2000, transportation activities account for 21.6% of the HKSAR Gross Domestic Product (GDP), which is equivalent to more than 200 billion Hong Kong dollars. Also, the contribution of transportation to the GDP has been increasing at a very high rate. Furthermore, the values listed here do not include the transportation activities of other service sectors. For example, manufacturers or retailers also require inbound and outbound logistics to support their daily operations even though these costs are not directly reflected in the proportion of transportation in the GDP. As commercial activities become global, logistics and transportation management cannot be neglected by any company in order to provide world-class service. The economic importance of transportation has motivated both private companies and academic researchers to use operation research and management science techniques to improve the efficiency of the transportation systems.

Various modes of transportation exist: including air, rail, ship and motor vehicles. The research on transportation has investigated different issues in each

mode. For air and rail, the focus is on the efficiency of the schedule of crews [41] and landing/dispatching control [6, 7, 40, 65, 71]. Comprehensive surveys include [2] for airline crew scheduling and [17] for train routing and scheduling. For ships and motor vehicles, the focus is on a common problem - the efficient use of a fleet of vehicles that must make a number of stops to pick up orders and/or deliver passengers or products. This kind of problem is referred as *Vehicle Routing Problem* (VRP). There are a huge amount of literature addressing different issues of this kind of problems and numerous well-demonstrated algorithms have been proposed to solve such problems. Interested readers may refer to the surveys [3, 10, 30].

In this thesis, only the land transportation will be discussed. More specifically, we focus on a map-based decision support system for land distribution and delivery planning in Hong Kong. As mentioned before, transportation related activities account for a significant amount of the GDP in Hong Kong. Moreover, the future growth is also predicted be considerable as the People's Republic of China enter the World Trade Organization (WTO) in January 2002. More amounts of goods/commodities will be exported from the Pearl River Delta of China through Hong Kong's container terminals, further emphasizing the economic importance of logistics management. Another motivation of developing this system and investigating the delivery planning in Hong Kong is the transportation difficulties in Hong Kong. The high variation of traffic conditions between the peak/non-peak hours in the region creates a lot of deviations to the delivery planning. In the following few lines, we try to highlight some of the distribution difficulties in Hong Kong and mention some the decision support tools used in our system.

Hong Kong is one of the busiest cities in the world. The total number of vehicles registered up to December 2000 was 517 thousand. Since the physical area of Hong Kong is only 1,098 square kilometers (km^2), it means that there are more than 450 vehicles per unit km^2 . The large number of vehicles leads to serious traffic congestion during the working and non-working hours. The congestion makes the actual vehicle-travel-time highly variable and difficult to predict. Usually, drivers or dispatchers are unable to adhere to the pre-assigned delivery routes due to traffic congestion problems. And eventually, this would lead to deviation from the original delivery plans and generate overtime costs or lower service levels due to late delivery. Traditional routing and scheduling techniques usually consider static situations. Such techniques may not be sufficient/suitable for the dynamic traffic environment like Hong Kong. Therefore, in our system, we considered another version of the routing and distribution planning such that the transit duration of a road segment is stochastic/time-dependent. This amendment to the general vehicle routing model is to take the variations of the traffic conditions into consideration. More precisely, two delivery planning models are considered. The first one is the *Time-varying Constrained Shortest Path* (TSCP) problem. The second one is the *Vehicle Routing Problem with Time Window and Stochastic Travel Times* (VRPTWST) problem. A literature review and the description of these two problems will be given in the next section.

1.2 Literature Review

Despite the dynamic nature of transportation networks, the majority of literature

on shortest path evaluation and vehicle routing problem has mainly concentrated on static problems. But as the dynamic information begins to play a more important role in decision-making problem, the focus of research has switched to dynamic and stochastic models in recent decades. Therefore, in this section, we classify problems into three types: Deterministic (static), stochastic (probabilistic) and dynamic (time-varying). In the following sub-sections, we will review the relevant research that has been reported in the literature about the two problems of the three different types. More detailed review will also be given in the section of introductory section in each chapter when we start to study the two problems.

1.2.1 Shortest Path Problem

A shortest path problem is stated as follows: *Given a network $N = (V, A, c)$, where V is the vertex set, A is the arc set, and $c(x, y)$ is the transit cost for traversing the arc $(x, y) \in A$, the problem is to find the cheapest path starting from a source vertex s to a sink vertex d .* The shortest path problems that have been studied in the literature can be classified into three classes. The first class of problems consider static version that all the parameters all deterministic. The second class of problems are stochastic in that the transit cost/duration for traversing an arc is random. The last class of problems consider dynamic version such that the transit cost/duration for traversing an arc is a function varying over time.

(1) Static problems

Shortest path problems were widely studied in the last half-century. Dijkstra [24] used a label setting approach to solve a single-source shortest path problem with nonnegative transit cost. Bellman [8] solved the same problem with negative transit cost by identifying negative cycles. Afterwards, many literatures follow

Dijkstra's idea and propose algorithms with improved time complexity [33, 36, 37]. Computational evaluations of these algorithms have been reported in [1, 14, 43, 78].

(2) Stochastic problems

The shortest path problem in stochastic, stationary networks is not as common as the previous static version. In this version, some of the parameters, such as transit cost or duration of the arc, follow a probability distribution (but not varying over time). Several papers have addressed the problem of this type. Frank [32] derived a closed form solution for the minimum travel time path through a stochastic, time-invariant network. A number of other works address similar problems [19, 47, 57] with different probability distributions of the transit cost/duration, and to determine the least expected time paths by setting the arc cost/time to its expected value and solving its equivalent deterministic problem. Moreover, more works presents algorithms for determining optimal paths in stochastic, time-invariant networks where the objectives are represented by utility function of various forms [28, 70].

(3) Dynamic problems

The dynamic shortest path problem is also called the time-varying shortest path problem. In this type of problem, the arc transit cost/duration is a deterministic function (discrete or continuous) of time so that the values vary with time. Cooke and Halsey [16] consider a discrete model in which the transit duration $b(x, y, t)$ varies as a function of the departure time t at vertex x , where waiting at vertex is not allowed. Similar problems are addressed by Orda and Rom [63]. Cai, Kloks and Wong [12] considering the relaxation the waiting at the vertex be arbitrary or

bounded by a given value. Sha [69] extends the algorithm to solve the k -shortest path problem.

Other research addresses shortest path problems in *Stochastic, Time-varying* (STV) networks. The arc cost/duration of these networks is a stochastic variable varying over time. Therefore, in STV, one cannot simply use the approach described in the stochastic version and solve the deterministic problem, see [59, 60].

1.2.2 Vehicle Routing Problem with Time Windows

Vehicle Routing Problem with Time Windows (VRPTW) is a generalized model of *Vehicle Routing Problem* (VRP). The VRPTW can be described as follows: *Given a network $G = (V_0, A)$, where V_0 is the set of vertices and A is the set of arcs. The same single depot is denoted by vertex 0 and vertex $n+1$, which represent the starting and the ending point of the routes respectively. Vertex 1, ..., n denote the vertices to be served and $V = V_0 \setminus \{0, n+1\}$. For each arc $(i, j) \in A$, $i \neq j$, there is an associated transit cost c_{ij} and transit duration t_{ij} . A set of vehicles K is located in the depot, and each of them possesses a limited capacity C_k , $k \in K$. At each vertex $i \in V$, it requires service within a specific time window $[a_i, b_i]$. If service must be made during the time window, it is referred as “hard” time window version. In contrast, if service can be made earlier or later than the time window, incurring penalty costs, it is referred to “soft” time-window version. The problem is to find a set of feasible vehicle-routes, originating and terminating at the single depot, with minimum traveling costs and penalty costs (if any).*

VRP and its variant problems have a lot of real-life applications. Therefore,

it arouses many interests of many researchers to study the problem. We divide the research reported in the literatures related to VRP into three categories: Static, stochastic and dynamic. The following sub-sections will review the problems of each kind.

(1) On the static problems

The static formulation is the most common model for the VRPTW. All the parameters in the formulation are deterministic. A lot of optimization-based and heuristic algorithms are proposed to solve the problem. They are generally divided into the following five classes:

- Branch-and-bound approaches [22, 23, 74]
- Cutting plane approaches [18, 29, 38, 46]
- Column generation approaches [52, 56]
- Heuristics and route improvement approaches [15, 31, 35, 51, 55, 62]
- Metaheuristics approaches [34, 64, 66, 76]

(2) On the stochastic problems

When some of the parameters of the VRP are random, the problem is referred as *Stochastic Vehicle Routing Problem* (SVRP). One extension considers the demand at each vertex to be random. The problem is to find a set of prior routes and restocking policies (if any) with minimum expected costs. Restocking policy here means the location/stock-level for the vehicle returning to the depot for replenishment. Two frameworks are considered, namely, chance-constrained [50] and recourse [5, 9, 25, 26, 27, 48].

Another extension investigates the problem that the transit duration between

any two vertices to be random. The problem is to find minimum expected travel costs without violating the duration limit of each route. Reported works of VRP with stochastic travel times are sparse. Only one paper [49] deals with this problem. Other related works include traveling salesman problem with probabilistic travel times [44, 53, 72].

(3) On the dynamic problems

Dynamic vehicle routing models focus on a single snap-shot of data. These models may be used in real-time to solve dynamic problems, or to develop vehicle schedules over the course of the day using data that is (assumed) known in advance. Deterministic dynamic vehicle routing problems arise in two settings: time-dependent demands, and time-dependent travel times.

Time-dependent demands appear generally when demand must be satisfied subject to specific time constraints. One case of this kind problem is the *Inventory Routing Problem* (IRP), where the determination of customer demands is based on the need to maintain customer inventories. A customer may require a delivery no later than day t and the service provider might decide to deliver earlier than day t if the cost is lower. The problem is to assign customers and deliveries to both days and vehicle routes. Related works include [41, 68].

The second class of problems considers time-dependent travel times. The only work that deals explicitly with vehicle routing and time-dependent travel time is [58]. In this paper, it assumes each link has a fixed travel time that is a function of the time a vehicle departs from the head of the link. The objective function is to minimize the total time required to complete all deliveries. A

nearest-neighbor tour construction heuristic is proposed.

1.3 Thesis Outline

The objective of this thesis is to conduct a study of routing and delivery planning in Hong Kong. We have developed a map-based decision support system dealing with this problem. This system provides map-based interface for the users to search, to update the company records such as customers, products, inventory and warehouses information.

At the same time, the system provides decision-support tools for route finding and delivery planning. Because of the highly varying traffic condition in Hong Kong, deterministic mathematical models and optimization techniques may not be sufficient to provide a good solution to the problem. Therefore, we have proposed one time-dependent and one stochastic model, named as *Time-varying Constrained Shortest Path at a 2-level Network* (TCSP2N) and *Vehicle Routing Problem with Time Windows and Stochastic Travel Times* (VRPTWST), and developed corresponding solution algorithms to each of them. These algorithms are also incorporated into the system so as to provide solutions to the problem with the consideration of the traffic conditions.

We outline the content of this thesis as follows: In Chapter 2, the TCSP2N together with two algorithms will be addressed. In Chapter 3, we will discuss about the formulation of the VRPTWST and propose an exact branch-and-cut algorithm to solve the problem. Afterwards, several techniques are considered to improve the solution times of the general algorithm. Some of the system features

and its implementation will be reported in Chapter 4. In Chapter 5, a brief survey on the commercial delivery planning software application will be given. Concluding remarks and further work will be discussed in Chapter 6.

Chapter 2

Time-Varying Shortest Path Problem with Constraints in a 2-Level Transportation Road Network

2.1 Introduction

Transportation network is highly dynamic. The transit time of a vehicle over a road segment varies over time due to traffic congestion or accidents. Therefore, using traditional shortest path model with fixed parameters may not be sufficient to represent the dynamic issues of transportation road network. Optimal routes solved by the traditional shortest path problem may not be the best solution under dynamic conditions. Because of this, in this chapter, we try to formulate the shortest path problem in a transportation road network as the Time-Varying Constrained Shortest Path Problem (TCSP).

The TCSP problem was first studied by Cai, Kloks and Wong [12]. In the model, they consider a set of vertices and arcs, with transit time and cost of each arc as a function of the time that the traversal begins at that arc. Moreover, postponement (i.e. waiting) of departure at a vertex may be allowed. The objective is to find the path with the least possible cost, with suitable waiting time at each node, subject to the constraint that the total traversal time of the path is at

most a given positive integer T . Three exact pseudo-polynomial algorithms are proposed for solving 3 different waiting scenarios, namely waiting at any vertex is forbidden, waiting at any vertex is arbitrarily allowed, and waiting at a vertex x is limited to an upper bound U_x .

The TSCP problem is a good foundation to model the shortest path problem in a transportation road network since it possesses the feature of varying transit time of each arc depending on the beginning time of the starting vertex of that arc. In this chapter, we try to extend the TSCP problem, with some transformations, to our 2-level road network. One exact algorithm and one heuristic algorithm will be presented.

We organize the reminder of this chapter as follows: In Section 2.2, some basic concepts and the problem formulation of the TSCP will be provided. In Section 2.3, the algorithm of solving the TSCP with arbitrary waiting time will be outlined. In Section 2.4, we will extend the TSCP to a 2-level network. Two algorithms for the 2-level problem will be described in Section 2.5. Concluding remarks will be given in Section 2.6.

2.2 Problem Formulation of TCSP

The formulation of the TSCP in a single level transportation road network is described as follows: Given a directed graph $G = (V, A)$, where V is the set of vertices and A is the set of the arcs. Let $b(x, y, u)$ and $c(x, y, u)$ be the transit time and the travel cost of the arc $(x, y) \in A$ respectively, which are functions of the beginning time u at the starting vertex x . Moreover, define $w(x, u)$ be the waiting

cost at the vertex x from time u to $u+1$, $T > u \geq 0$. We assume that $b(x, y, u)$, $c(x, y, u)$ are positive for all $(x, y) \in A$. In this problem, we let $n = |V|$, $m = |A|$ and u is indexed as $0, 1, \dots, T$.

Definition 2.1 *A waiting time $\omega(x)$ at a vertex x is a nonnegative integer, and $U_x \geq 0$ is its upper bound.*

Definition 2.2 *Define $P = (s = x_1, \dots, d = x_r)$ be a path from s to d . Let $\omega(x_i)$, $i = 1, \dots, r$, be the waiting times at vertices x_1, \dots, x_r . Let*

$$\tau(x_1) = \omega(x_1) \text{ and}$$

$$\tau(x_i) = \omega(x_i) + \tau(x_{i-1}) + b(x_{i-1}, x_i, \tau(x_{i-1})) \text{ for } i = 2, \dots, r,$$

where $\tau(x_i)$, $i = 1, \dots, r$, is defined as the departure time at vertex x_i along P .

Definition 2.2 is the relation of the departure time of the vertex along the path. With this relation, a dynamic programming updating approach is proposed by Cai, Kolks & Wong [12]. One of the algorithms with arbitrary waiting time at any vertex will be outlined to give a fundamental idea in developing algorithms in a 2-level network.

2.3 Arbitrary Time

In this section, we examine the problem when there is no constraint imposed on the waiting time of each vertex. Assume $b(x, y, u)$ is positive. Let $d_A(y, t)$ be the cost of the cheapest path from a vertex s to another vertex y of time at most t , where waiting time at any vertex is not restricted. This problem is referred as *Time-Varying Constrained Shortest Path with Arbitrary Waiting Time*

(TCSP-AWT).

We use the following recursive relation to compute $d_A(y, t)$. Note that the optimal waiting times of each vertex can be obtained implicitly by the recursive computations.

Lemma 2.1 $d_A(s, t) = 0$ for all t and $d_A(y, 0) = \infty$ for all $y \neq s$. For $t > 0$ and $y \neq s$, we have

$$d_A(y, t) = \min \left\{ d_A(y, t-1) + w(y, t-1), \min_{\{x | (x, y) \in A\}} \min_{\{u | u + b(x, y, u) = t\}} \{d_A(x, u) + c(x, y, u)\} \right\}$$

The proof of Lemma 2.1 is in [12].

Definition 2.3 For every arc $(x, y) \in A$ and for $t = 0, \dots, T$, let

$$\gamma_A(x, y, t) = \min_{\{u | u + b(x, y, u) = t\}} \{d_A(x, u) + c(x, y, u)\}.$$

We adopt the convention that $\gamma_A(x, y, t) = \infty$ whenever $\{u | u + b(x, y, u) = t\} = \emptyset$.

The result follows directly from Lemma 2.1.

Corollary 2.1

$$d_A(y, t) = \min \left\{ d_A(y, t-1) + w(y, t), \min_{\{x | (x, y) \in A\}} \gamma_A(x, y, t) \right\}.$$

With Lemma 2.1, we describe a dynamic programming algorithm to solve TCSP-AWT. From Corollary 2.1, if we update $d_A(y, t)$, we have to know $\gamma_A(x, y, t)$ for all $(x, y) \in A$. Given t and (x, y) , $\gamma_A(x, y, t)$ could be evaluated by enumerating $0 \leq u \leq T$ to find those satisfying $u + b(x, y, u)$, according to

Definition 2.2. However, this would require a worst-case running time $O(T)$ for every t . Therefore, [12] suggests to first sort the values of $u + b(x, y, u)$ for all $u = 1, 2, \dots, T$ and all arcs $(x, y) \in A$, before computing the recursive relation of $d_A(y, t)$ given in Lemma 2.1.

Algorithm 2.1

Begin

Initialize $d_A(s, t) = 0$ and $\forall_x \neq s \ d_A(x, 0) = \infty$ for $t = 0, \dots, T$

Sort all values $u + b(x, y, u)$ for $u = 1, \dots, T$ and for all arcs $(x, y) \in A$

For $t = 0, \dots, T$ **do**

For every arc $(x, y) \in A$ **do** $\gamma_A(x, y, t) := \infty$

For all arcs $(x, y) \in A$ and all u such that $u + b(x, y, u) = t$ **do**

$\gamma_A(x, y, t) := \min \{ \gamma_A(x, y, t), d_A(x, u) + c(x, y, u) \}$

For every vertex y **do**

$d_A(y, t) := \min \{ d_A(y, t-1) + w(y, t), \min_{(x, y) \in E} \gamma_A(x, y, t) \}$

End

Theorem 2.1 *The TCSP-AWT problem with positive transit times can be optimally solved in $O(T(n + m))$.*

Detailed proof of Theorem 2.1 is given in [12]. However, it is easy to trace out from Algorithm 2.1.

Algorithm 2.1 computes the shortest length $d_A(x, T)$ of the shortest path from a origin vertex s to x within a given time limit T . In order to identify the waiting time at each vertex along the path, we can use a standard backtracking procedure

of dynamic programming. Let the shortest path be $P^* = (s = x_1^*, \dots, x_i^*, x_j^*, \dots, x_r^*)$, then the waiting times of each vertex can be obtained from the departure times of the vertices. For example, if $\tau(x_i^*)$ and $\tau(x_j^*)$ are the optimal departure times at the two vertices x_i^* and x_j^* , then the optimal waiting time at the vertex x_j^* is

$$w_i(x_j^*) = \tau(x_j^*) - \tau(x_i^*) - b(x_i^*, x_j^*, \tau(x_i^*)).$$

In [12], there are 2 more algorithms solving the scenarios when the waiting time is restricted as 0 or bounded by a positive value. Because the similarity of these algorithms to Algorithm 2.1, we are not going to discuss them in detail. In next section, we try to extend the TCSP at a single level network to a 2-level network using the idea given in this section.

2.4 TCSP at 2-level Network

Multi-level network is widely used in daily life. The use of multi-level network to replace a single-level complicated network can decrease the computational effort in solving network optimization problem. By dividing a single-level network, with a huge number of vertices and arcs, into a number of smaller and simpler sub-networks, the complexity of computation at each sub-network can be reduced.

Examples of multi-level network include delivery systems, transportations, telecommunications and computer networks. For instance, in United Parcel Service (UPS), operations are broken down into many levels. From the start of the pickup process, parcels are collected by vans or motorcycles and centralized

at some shuttle points. Then, bigger trucks are used to collect the parcels from different shuttle points and send to a handling hub. All parcels from the region will be concentrated and deliver through airplanes. Similar network appears in the supply chain model that a plant delivers goods to some warehouses. Then, the warehouses send the goods to the distribution centers located at different regions.

In this section, we try to model the transportation road network in Hong Kong into 2 levels, namely the *Main Road Level* (MRL) and the *Sub-Street Level* (SSL). We propose one exact algorithm and one heuristic algorithm to find the TCSP, starting from an origin vertex at any levels and arriving at a destination vertex at any levels within a given time limit T . Also, we only consider the scenario that waiting time at any vertex is arbitrary.

2.4.1 Problem Formulation of TCSP at 2-level Network

Consider $G_i = (V_i, A_i)$, $i = 0, 1, \dots, K$, as $K + 1$ directed graphs, with V_i be the set of vertices and A_i be the set of arcs of G_i respectively. In our model, G_0 is defined as the *Main Road Layer* (MRL) and G_i , $i = 1, \dots, K$, are defined as the *Sub-Street Level* (SSL).

Each graph in the SSL is connected with MRL with directed arcs. Define R_i , $i = 1, \dots, K$, be a set of directed arcs (x, y) connecting from any node $x \in V_i$, $i = 1, \dots, K$, to any node $y \in V_0$. Also, define R_0 be the set of the directed arcs (y', x') connecting from any node $y' \in V_0$ to any node $x \in V_1 \cup \dots \cup V_K$. Figure 3.1 shows the layout of the network.

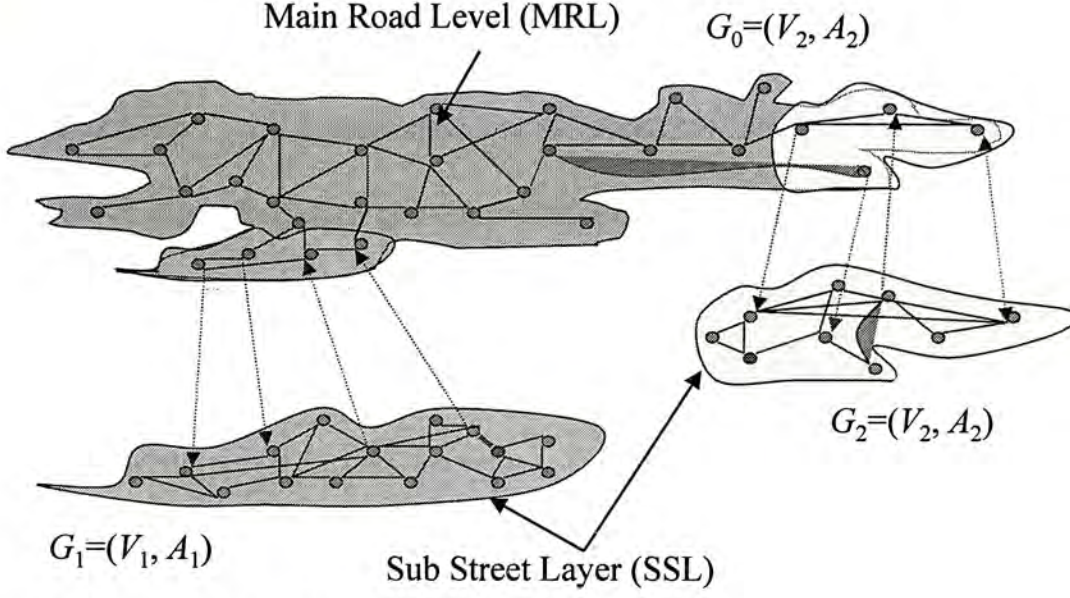


Figure 2.1 Layout of 2-layer network

Let $b(x, y, u)$ be the transit time needed to traverse an arc $(x, y) \in A_i$, $i = 0, 1, \dots, K$ and $c(x, y, u)$ be the cost to traverse the arc. Both of the $b(x, y, u)$ and $c(x, y, u)$ are functions of the departure time u at the starting vertex x , where $u \in [0, T]$, $T > 0$ is a given integer, which is the maximum time that is allowed to traverse a whole path. Moreover, let $w(x, u)$ be the waiting cost at the vertex x from time $u - 1$ to u , $u \geq 1$. Also, let $f(x, y, u)$ be the scaling factor to the transit time depending on time u , $d(x, y)$ and $s(x, y)$ be the length and the average traversal speed of the arc $(x, y) \in A_i$. We assume that $b(x, y, u)$ is positive, $c(x, y, u)$, $d(x, y)$ and $s(x, y)$ are nonnegative and $f(x, y, u) \geq 1$. Throughout this section, we let $n_i = |V_i|$, $m_i = |E_i|$ and $r_i = |R_i|$ for all $i = 0, \dots, K$.

For implementation, $c(x, y, u)$ and $b(x, y, u)$ are defined as follows:

The *cost* of a path in our problem is defined as a linear combination of the travel distance cost and travel time cost. As the cost of a vehicle route, it is reasonable to assume that some operating costs are distance related, say fuel and maintenance cost. And some costs are time related such as the salary of drivers.

Thus, we define the transit cost of an arc in the following manner:

$$c(x, y, u) = d(x, y) \times C_D + b(x, y, u) \times C_T \quad (2.1)$$

where C_D is the travel cost per unit of distance and C_T the travel cost per unit of time. That means the cost of traversing an arc is a weighted sum of the distance travelled and the time spent on that arc. With this formulation, the shortest path problem can also be stated as the cheapest path problem in the sense that the total cost of traversing the path is minimized. We would use the terms “shortest path” and “cheapest path” interchangeably throughout this chapter.

The other issue of our problem is how to model the time-varying factor in the road network. In a road network, the variation of the transit time is due to the different number of vehicles using the road during different intervals of time. Or, accident occurs at some of the road segments at a particular point of time so that these road segments are blocked or congested. Moreover, geographical location may also affect the transit time of the road segment. It is obviously that the road segments in the urban area are usually more congested than those in the rural area. Because different pieces of road segment at different points of time may have different transit times, we try to introduce a scaling factor to $f(x, y, u)$ to measure the degree of congestion of an arc (x, y) at the time t . In our formulation, $f(x, y, u)$ is a step function of time u . It is because we assume that u is discrete and the traffic condition does not change during the time interval that is represented by the time index u . It is a reasonable assumption if the length of time represented by a single index is short, which is in the sense that the traffic condition is unlikely to change explicitly during this short period of time. Then, the transit time of an arc (x, y) with departure beginning at time u is defined as follow:

$$b(x, y, u) = f(x, y, u) \times d(x, y) / s(x, y) \quad (2.2)$$

If $f(x, y, u)$ equals to 1, that means the road segment (x, y) is in normal condition such that the transit time is just the length divided by the average speed. If $f(x, y, u)$ is larger, that means the arc is more congested and thus the transit time will become larger.

Moreover, to simplify the complexity of our problem, each graph in the SSL is not directly connected to another. That means finding a path between 2 different graphs at the SSL, the path must pass through the MRL by traversing those arcs in $R_i, i = 0, \dots, K$. Also, arcs in $R_i, i = 0, \dots, K$, perform as dummy arcs connecting MRL and SSL. Since zero transit time would lead to more complicated computations (see [12] for details), we would let $d(x, y) = 0$ and $b(x, y, u) = 1$, for all $(x, y) \in R_i, i = 0, \dots, K$.

This 2-level network is to model the transportation road network in Hong Kong, named as *2-level Transportation Road Network* (2LTRN). In a real transportation road network, districts are connected by highways, expresses or tunnels. These road segments correspond to those arcs in the MRL. Inside each district, there are many streets or small avenues, which correspond to the arcs in each graph in the SSL. Vertices in the network represent the junctions of the road segments.

2.5 Algorithms for Solving TCSP in a 2-level Network

The objective of the *Time-Varying Constrained Shortest Path Problem in a 2-Level Network* (TCSP2N) is to find a minimum *cost* path from an origin vertex

s at any level to a destination vertex d at any level. The *cost*, as defined in Equation 2.1, is a weighted sum of the travelling costs in terms of distances and times, and the waiting costs at each vertex along the path.

2.5.1 Exact Algorithm

If we ignore the *level* concept in our model, the TCSP2N can also be considered as a TCSP at a single network given in Section 2.2, which is to find the cheapest path, in a single-level time-varying network, from an origin vertex at time zero and arrive to a destination before a given time limit T .

With suitable transformation, our 2-level network can be modeled as one entire directed graph $G = (V, E)$ with $V = V_0 \cup \dots \cup V_K$ and $A = A_0 \cup \dots \cup A_K \cup R_0 \cup \dots \cup R_K$. Then, we can let $n' = |V| = \sum_{i=0}^K n_i$ and $m' = |A| = \sum_{i=0}^K m_i + \sum_{i=0}^K r_i$. Then, we can apply Algorithm 2.1 to the transformed network for arbitrary waiting time. For the time complexity of the transformation, by adding new arcs and vertices to the distance matrix and order table, we need $O(n' + m')$ steps. Combine this with Theorem 2.1, we come up with the following theorem:

Theorem 2.2 *The TCSP2N with arbitrary waiting time at each vertex and positive transit times of each arcs can be optimally solved in $O(T(n' + m'))$.*

However, recall the assumption of 2-level network that each graph in the SSL is not directly connected each other. Then, TCSP2N with arbitrary waiting time can be divided into 5 possible scenarios:

1. from any vertices at the MRL to any other vertices at the MRL, referred as $\text{MRL} \rightarrow \text{MRL}$,
2. from any vertices at one of graphs in the SSL to any other vertices at the same graph in the SSL, referred as $\text{same SSL} \rightarrow \text{same SSL}$,
3. from any vertices at the MRL to any vertices at one of the SSL, referred as $\text{MRL} \rightarrow \text{SSL}$,
4. from any vertices at one of the of graphs in the SSL to any vertices at the MRL, referred as $\text{SSL} \rightarrow \text{MRL}$, and
5. from any vertices at one of the graphs in the SSL to any vertices at another graph in the SSL, referred as $\text{diff. SSL} \rightarrow \text{diff. SSL}$.

By breaking down the TCSP2N into several sub-problems, we then do not require to expand the entire network. By this way, we only need to expand the related graphs in the SSL and the MRL.

Definition 2.4 *If the source vertex s of the TCSP2N is from one of the graphs in the SSL, say $G_k = (V_k, A_k)$, then denote $G_k = G_S = (V_S, A_S)$ and $R_k = R_S$. Also, we have $n_S = |V_S|$, $m_S = |A_S|$, $r_S = |R_S|$ and $s \in V_S$. G_S is called as the source graph (SG).*

Definition 2.5 *If the destination vertex d of the TSCP2N is from one of the graphs in the SSL, say $G_l = (V_l, A_l)$, then denote $G_l = G_D = (V_D, A_D)$ and we have $n_D = |V_D|$, $m_D = |A_D|$ and $d \in V_D$. G_D is called as the destination graph (DG).*

Definition 2.6 *If G_D is the destination graph and is one of the graph in the SSL,*

define $R_D = \{(x, y) \in R_0 \mid y \in V_D\}$ with $|R_D| = r_D$.

Using the idea of expanding 2-level network into a single graph, we can apply Algorithm 2.1 to the expansions of the 5 scenarios.

For $MRL \rightarrow MRL$ and same $SSL \rightarrow$ same SSL , expansion is not required and Algorithm 2.1 can be directly applied with time complexity $O(T(n_0 + m_0))$ and $O(T(n_k + m_k))$, for $k = 1, \dots, K$, respectively.

For $MRL \rightarrow SSL$, combine G_0 , G_D and R_D together and apply the algorithm with time complexity $O(T(n_0 + n_D + m_0 + m_D + r_D))$. Similarly, for $SSL \rightarrow MRL$, combine G_0 , G_S and R_S together. Then, the complexity is $O(T(n_0 + n_S + m_0 + m_S + r_S))$.

At last, for diff. $SSL \rightarrow$ diff. SSL , we need to combine G_0 , G_D , R_D , G_S and R_S together so that the time complexity of the algorithm is $O(T(n_0 + n_S + n_D + m_0 + m_S + m_D + r_S + r_D))$.

Theorem 2.3 *The TCSP-2TRN problem with suitable expansion can be optimally solved in $O(T(n_0 + n_S + n_D + m_0 + m_S + m_D + r_S + r_D))$.*

2.5.2 Heuristic Algorithm

In this section, a solution heuristic will be suggested for the TCSP-2N. The idea of the heuristic is to divide the whole problem into the three subproblems, and solve each subproblem independently. But before presenting the idea of the heuristic, we need to make the following assumptions.

Assumption 2.1 *The number of vertices and arcs of the MRL are much greater than that of all graphs in the SSL, that is $n_0 \gg n_i$ and $m_0 \gg m_i$, for all $i = 1, \dots, K$.*

Assumption 2.2 *The physical area covered by the MRL is much greater than that in all graphs in the SSL, that is for all $i = 1, \dots, K$, we have*

$$\sum_{(x_1, y_1) \in E_0} d(x_1, y_1) \gg \sum_{(x_2, y_2) \in E_i} d(x_2, y_2)$$

In the following analysis, we only consider the worst case, which is diff. SSL \rightarrow diff. SSL, as further consideration. Then, the problem becomes to find the path with minimum cost from a source vertex $\in V_S$ to a destination vertex $y \in V_D$ starting from time zero with a given time limit T .

Definition 2.7 *If G_S is the source graph and $s \in V_S$ is the source vertex, let T_S be the time required to traverse a path from s to $q_S \in V_S$ such that $(q_S, x_S) \in R_S$ and $x_S \in V_0$. q_S is named as Destination of the Source Graph (D-SG).*

Definition 2.8 *If G_D is the destination graph and $y \in V_D$ is the destination vertex, let T_D be the time required to traverse a path from $q_D \in V_D$ to y such that $(x_D, q_D) \in R_D$ and $x_D \in V_0$. q_D is named as Origin of the Destination Graph (O-DG).*

Definition 2.9 *For x_S and x_D as defined in Definition 3.4 and 3.5 respectively, let T_0 be time required to traverse the path from x_S to x_D in the MRL. x_S and x_D are*

named as O-MRL and D-MRL respectively.

One characteristic of our model is that the *size* of the MRL is much larger than that of all graphs in the SSL. The *size* here is in term of physical area covered by the graph. From Assumption 3.1 and 3.2, if one would like to find a path from one graph in the SSL to another graph in the SSL through MRL, we can claim that it is more likely to spend more time in the MRL than the two graphs in the SSL. It is a sensible assumption that the MRL covers the area of whole Hong Kong and each graph in the SSL only covers a small region.

Assumption 2.3 *For diff. SSL \rightarrow diff. SSL, the time required to traverse the path in the MRL, T_0 is much greater than the time required to traverse the path in the source graph, T_S and the destination graph, T_D .*

By Assumption 2.3, we know that the time required of traversing the MRL dominates the others. Since the objective function is a linear combination of the distance, time and the waiting times, that the *cost* is dominated by the path in MRL. Then, we can come up with our solution heuristic by solving the TCSP in the MRL from O-MRL to D-MRL first, and follow by solving the TSCP in the source graph (SG) and the destination graph (DG).

Definition 2.10 *Let α_0 be a given proportion to the time limit T in order to traverse the path from O-MRL to D-MRL in the MRL. Also, let α_S and α_D be the given proportion to the time limit T in order to traverse a path from s to D-SG and O-DG to y respectively such that*

$$\alpha_0 + \alpha_S + \alpha_D = 1, \text{ and } 0 < \alpha_S, \alpha_D \ll \alpha_0 < 1.$$

In our algorithm, the TCSP is solved in the MRL first, named as the *Middle Problem* (MP). However in real situation, the path starts from the SG and goes to the MRL then. Certain values of time are kept for the path in SG and DG before solving the TCSP-TRN in the MRL. Therefore, our MP is the find the path with minimum cost starting from the node x_S (O-MRL) to x_D (D-MRL), with time starting at $\alpha_S T$ and finishing before $T(1-\alpha_D)$. Define $d_{MP}(x, t)$ be the cost of the cheapest path from the starting vertex x_S to another vertex $x \in V_0$ of time at most t , $t = \alpha_S T, \alpha_S T + 1, \dots, T(1-\alpha_D) - 1, T(1-\alpha_D)$, where waiting at any vertex is not restricted. Then, the following recursive relation directly follows from Lemma 2.1.

Lemma 2.2 $d_{MP}(x_S, t) = 0$ for all $t = \alpha_S T, \alpha_S T + 1, \dots, T(1-\alpha_D) - 1, T(1-\alpha_D)$ and $d_A(x, \alpha_S T) = \infty$ for all $x \neq x_S$. For $\alpha_S T < t \leq T(1-\alpha_D)$ and $x \neq x_S$, we have

$$d_{MP}(x, t) = \min \left\{ d_{MP}(x, t-1) + w(x, t), \min_{\{x' | (x', x) \in A_0\}} \min_{\{u | u + b(x', x, u) = t\}} \{d_{MP}(x', u) + c(x', x, u)\} \right\}$$

Definition 2.11 For every arc $(x', x) \in A_0$ and for $t = \alpha_S T, \dots, T(1-\alpha_D)$, let

$$\gamma_{MP}(x', x, t) = \min_{\{u | u + b(x', x, u) = t\}} \{d_{MP}(x', u) + c(x', x, u)\}.$$

Let $\gamma_{MP}(x', x, t) = \infty$ whenever $\{u | u + b(x', x, u) = t\} = \emptyset$.

Definition 2.12 Let $P_0^* = (x_1^* = x_S, \dots, x_r^* = x_D)$ be the optimal path in the MRL.

Define t_S^* and t_D^* be the optimal departure time at the vertex x_S and the optimal

arrival time to the vertex x_D under the optimal path P_0^* .

For convenience, we regard the TCSP problem at the source graph and at the destination graph as *Left Problem* (LP) and *Right Problem* (RP) respectively. The LP is to find to the shortest path starting from the origin vertex s to q_S (D-SG), with time starting at zero to t_S^* . Similarly, the RP is to find the shortest path starting from q_D (O-SG) to the destination vertex d , with time starting from t_D^* to T .

Lemma 2.3 $d_{LP}(x_S, t) = 0$ for all $t = 0, \dots, t_S^*$ and $d_{LP}(x, 0) = \infty$ for all $x \neq x_S$.

For $0 < t \leq t_S^*$ and $x \neq x_S$, we have

$$d_{LP}(x, t) = \min \{ d_{LP}(x, t-1) + w(x, t), \min_{\{x' | (x', x) \in A_S\}} \min_{\{u | u + b(x', x, u) = t\}} \{ d_{LP}(x', u) + c(x', x, u) \} \}$$

Definition 2.13 For every arc $(x', x) \in A_S$ and for $t = 0, \dots, t_S^*$, let

$$\gamma_{LP}(x', x, t) = \min_{\{u | u + b(x', x, u) = t\}} \{ d_{LP}(x', u) + c(x', x, u) \}.$$

Let $\gamma_{LP}(x', x, t) = \infty$ whenever $\{u | u + b(x', x, u) = t\} = \emptyset$.

Lemma 2.4 $d_{RP}(x_D, t) = 0$ for all $t = t_D^*, \dots, T$ and $d_{RP}(x, t_D^*) = \infty$ for all $x \neq x_D$.

For $t_D^* < t \leq T$ and $x \neq x_D$, we have

$$d_{RP}(x, t) = \min \{ d_{RP}(x, t-1) + w(x, t), \min_{\{x' | (x', x) \in A_D\}} \min_{\{u | u + b(x', x, u) = t\}} \{ d_{RP}(x', u) + c(x', x, u) \} \}$$

Definition 2.14 For every arc $(x', x) \in A_D$ and for $t = t_D^*, \dots, T$, let

$$\gamma_{RP}(x', x, t) = \min_{\{u \mid u + b(x', x, u) = t\}} \{d_{RP}(x', x, t) + c(x', x, u)\}.$$

Let $\gamma_{RP}(x', x, t) = \infty$ whenever $\{u \mid u + b(x', x, u) = t\} = \emptyset$.

Algorithm 2.2

Begin

$opt-cost := \infty$

For every arc $(q_S, x_S) \in R_S$ **do**

For every arc $(x_D, q_D) \in R_D$ **do**

Initialize $d_{MP}(x_S, t) = 0$ and $\forall_{x \neq x_S, x \in V_0} d_{MP}(x, t) = \infty$ for $t = \alpha_S T, \dots,$

$T(1 - \alpha_D)$

Sort all values $u + b(x, y, u)$ for $u = \alpha_S T + 1, \dots, T(1 - \alpha_D)$ for all arcs $(x, y) \in A_0$

For $t = \alpha_S T, \dots, T(1 - \alpha_D)$ **do**

For every arc $(x, y) \in A_0$ **do** $\gamma_{MP}(x, y, t) := \infty$

For all arcs $(x, y) \in A_0$ and all u such that $u + b(x, y, u) = t$ **do**

$$\gamma_{MP}(x, y, t) := \min \{ \gamma_{MP}(x, y, t), d_{MP}(x, u) + c(x, y, u) \}$$

For every vertex y **do**

$$d_{MP}(y, t) := \min \{ d_{MP}(y, t - 1) + w(y, t), \min_{(x, y) \in E_0} \gamma_{MP}(x, y, t) \}$$

$$d_{MP}^*(x_D, t) = \min_t d_{MP}(x_D, t)$$

Set t_D^* be the optimal arrival time to x_D

Set t_S^* be the optimal leaving time from x_S

Initialize $d_{LP}(s, t) = 0$ and $\forall_{x \neq s, x \in V_S} d_{LP}(x, t) = \infty$ for $t = 0, \dots, t_S^*$

Sort all values $u + b(x, y, u)$ for $u = 1, \dots, t_S^*$ for all arcs $(x, y) \in E_S$

For $t = 0, \dots, t_S^*$ **do**

For every arc $(x, y) \in A_S$ **do** $\gamma_{LP}(x, y, t) := \infty$

For all arcs $(x, y) \in A_S$ and all u such that $u + b(x, y, u) = t$ **do**

$$\gamma_{LP}(x, y, t) := \min \{ \gamma_{LP}(x, y, t), d_{LP}(x, u) + c(x, y, u) \}$$

For every vertex y **do**

$$d_{LP}(y, t) := \min \{ d_{LP}(y, t-1) + w(y, t), \min_{(x,y) \in E_0} \gamma_{LP}(x, y, t) \}$$

$$d_{LP}^*(q_S, t) = \min_t d_{LP}(q_S, t)$$

Initialize $d_{RP}(q_D, t) = 0$ and $\forall_{x \neq q_D, x \in V_D} d_{RP}(x, t) = \infty$ for $t = t_D^*, \dots, T$

Sort all values $u + b(x, y, u)$ for $u = t_D^* + 1, \dots, T$ for all arcs $(x, y) \in E_D$

For $t = t_D^*, \dots, T$ **do**

For every arc $(x, y) \in A_D$ **do** $\gamma_{RP}(x, y, t) := \infty$

For all arcs $(x, y) \in A_D$ and all u such that $u + b(x, y, u) = t$ **do**

$$\gamma_{RP}(x, y, t) := \min \{ \gamma_{RP}(x, y, t), d_{RP}(x, u) + c(x, y, u) \}$$

For every vertex y **do**

$$d_{RP}(y, t) := \min \{ d_{RP}(y, t-1) + w(y, t), \min_{(x,y) \in E_D} \gamma_{RP}(x, y, t) \}$$

$$d_{RP}^*(d, t) = \min_t d_{RP}(d, t)$$

If $d_{MP}^*(x_D, t) + d_{LP}^*(q_S, t) + d_{RP}^*(d, t) \leq \text{opt-cost}$ **Then**

$$\text{opt-cost} := d_{MP}^*(x_D, t) + d_{LP}^*(q_S, t) + d_{RP}^*(d, t)$$

End

Algorithm 2.2 gives a heuristic pseudo-polynomial time solution to the TSCP2N. The idea of Algorithm 2.2 is solving the MP first by giving 2

parameters α_S and α_D . Then, we solve the TCSP of the MP from time $t = \alpha_S T$ to $T(1-\alpha_D)$. After finding the optimal path of the MP, the arrival time t_S^* and departure time t_D^* of D-MRL and O-MRL can be identified. The algorithm follows by solving TCSP of the LP and RP with corresponding times and vertices in the source graph and the destination graph.

Since the MP is the TCSP in the MRL, the numbers of nodes and arcs are n_0 and m_0 respectively. The length of the time is $T(1-\alpha_S-\alpha_D)$. Therefore, the MP can be solved in $O((1-\alpha_S-\alpha_D)T(n_0+m_0))$. Similarly, the LP and RP can be solved in $O(T(n_S+m_S))$ and $O(T(n_D+m_D)n_D)$. Therefore, the complexity of Algorithm 2.2 is $O((1-\alpha_S-\alpha_D)T(n_0+m_0)) + O(T(n_S+m_S)) + O(Tn_D(n_D+m_D))$.

2.6 Concluding Remarks

In this chapter, we have considered an extension of the Time-varying Constrained Shortest Path from a single-level network to a 2-level network. We use the algorithm proposed by Cai, Kloks & Wong [11] at single-level network as a basic algorithm solving the TCSP in 2-level network. The importance of our work is to reduce the time complexity when apply the original algorithm in a large and complicated network. By breaking down a one-layer network logically into 2-level, we can ignore some of the unrelated vertices and arcs when the solving the problem, hence improving the solution time. A logical division of the transportation road network is presented. With the characteristics of this 2-level network model, a heuristic algorithm is proposed.

To conclude, we address the applications of the TCSP calculation. The applications can be divided into two aspects. With real time information updating

of the scaling factor $f(x, y, u)$ according to the degree of congestion of the road segments, the TCSP can provide a good suggestion to the drivers of finding the path between two locations. With historical data, TCSP can still use as a backbone of the delivery problem in estimating the travel times between two locations at different times of a day. Since general delivery scheduling problem usually use a constant value to represent the travel time between two locations, it may not be sufficient to reflect the real situation of a one-day long delivery schedule.

Chapter 3

Vehicle Routing Problem with Time Windows and Stochastic Travel Times

3.1 Introduction

Designing optimal delivery or collection routes, where vehicles originate from a single depot and visit a number of stops, is generally referred to as the *Vehicle Routing Problem* (VRP). When solving the VRP in real applications, it is common practice to assume that the travel times along the arcs are known. This kind of problem is deterministic, and many algorithms have been developed to deal with it. Such methods have been applied extensively in practice. However, in many real-world applications, one or more parameters of VRP tend to be random. For example, the set of customers to be served, the customer demand, or the travel times can be stochastic. These problems are referred as Stochastic Vehicle Routing Problem (SVRP). In this chapter, we consider the SVRP where the travel times between two points are stochastic, and also each customer requires service within a specific time window. All other data is deterministic. This problem is referred as *Vehicle Routing Problem with Time Windows and Stochastic Travel Times* (VRPTWST).

One of the practical applications of this problem is distribution planning of

logistic companies and supermarkets. Customers may require delivery during preferred time slots so that the planner needs to find a set of routes to satisfy each order within a particular time slot while minimizing the total operating costs. Another application is repairing team scheduling of telecommunication, electricity or town gas providers. Services are required at different times and an optimal-cost schedule can be obtained for each repairman. The stochastic travel time issue is significant if the traffic condition varies highly over time. This situation always appears in some busy cities such as Hong Kong, Shanghai, Singapore, New York or Paris. The variation in the travel times may lead to situations where some customers cannot receive service at their chosen times, and hence may generate losses to the company. Therefore, finding a delivery schedule while minimizing a combination of the travel costs and expected losses under stochastic travel conditions is useful in distribution planning in many cities with busy traffic.

Related works of Vehicle routing problem with stochastic travel time are sparse. Laporte, Louveaux and Mecure [49] first proposed the formulation of finding the optimal vehicle routes of visiting several points, where the travel time between any 2 points is a stochastic variable. Three models are proposed. The first model is to find the vehicle routes with least operating cost while ensuring that the probability of the duration of a route exceeding a specific length, B , is not greater a given value α . This model is referred as the chance constrained model. The second and the third model are recourse models. The objective is to find the vehicle routes with least operating costs together with the expected penalty when the duration of the route is exceeding B . The difference between the second model and the third model is that the third model does not identify which vehicle

to serve which route. But it requires the assumption that all vehicles are homogenous. Our work mainly extends the second model and formulates a similar problem but with time-window requirement at each vertex.

We organize the remainder of this chapter as follows. Section 3.2 will give the problem formulation. Section 3.3 will propose a general branch-and-cut algorithm. Some issues to improve to the solution time of the algorithm will be discussed in Section 3.4. Afterwards, a modified branch-and-cut algorithm will be introduced in Section 3.5. Computational comparisons and analysis will be given in Section 3.6. Finally, conclusion will be given in Section 3.7.

4.2 Problem Formulation

In this section, we will give the formulation of the problem. The formulation is an extension of Laporte, Louveaux and Mecure [49]. Consider a directed graph $G = (V_0, A)$ where V_0 is the set of nodes and A is the set of the arcs. Vertex 0 and vertex $n+1$ denote the same single depot representing the origin and the destination point of the routes respectively. Vertices 1, ..., n denote the customers to be visited. Also, $V = V_0 \setminus \{0, n+1\}$ and $A = \{(i, j): i \neq j, \forall i \in V \cup \{0\}, \forall j \in V \cup \{n+1\}, i \neq j\}$. Inside the depot, there is a set of vehicles available and denoted by K .

For each arc $(i, j) \in A$, there is an associated distance or travel cost c_{ij} (may be different between vehicles), which may represent the cost of fuel and maintenance in practice. Also, a stochastic travel time t_{ij} is associated with A . This travel time is assumed to be a discrete random variable in our study. For

example, the state of traffic conditions can be peak, off-peak or accident, which may have different impacts on the travel times.

For each vertex $i \in V$, customer has demand D_i , and service must be made within a time window $[a_i, b_i]$. Otherwise, a unit penalty cost β_i^- will be incurred per unit of time as the loss of the reputation or service level to the company. In contrast, if a vehicle k arrives earlier than the time window, it may need to wait until service can be provided. Therefore, in this case, a unit waiting cost β_k^+ will be paid for the parking fee of the vehicle or other costs to the driver. This kind of time window is referred to as *soft time windows*. Moreover, we assume that all vertices are restricted to be served by exactly one vehicle.

There is a set of vehicles K located in the depot. For each vehicle $k \in K$, a set-up cost f_k is needed if it is used to serve any route. Moreover, it has a capacity limit C_k (assumed to be in the same unit measure as the demand), and also it has working duration $[d_k, e_k]$. If the vehicle working exceeds the working duration, an overtime cost γ_k per unit of time will be incurred.

A three-index simple recourse model can be defined for the problem. The first stage problem is to decide the number of vehicles used and their routes. Travel time of the arcs can be realized afterwards. The second stage problem is to find the optimal arrival time at each vertex such that the total of waiting cost, late delivery penalty and overtime cost is minimized. Define

f_k = the fixed set-up cost for vehicle k ;

C_k = the capacity limit of vehicle k ;

$[d_k, e_k]$ = the normal working time of vehicle k ;

γ_k = the overtime cost per unit of time of vehicle k ;

β_k^+ = the waiting cost per unit of time of vehicle k ;

D_i = the demand of vertex i ;

$[a_i, b_i]$ = the time window of vertex i allowed to be served without penalty;

β_i^- = unit penalty cost per unit of time for late delivery of vertex i ;

τ_{ik} = the service time of vertex i by vehicle k ;

c_{ijk} = travel cost of vehicle k on the arc (i, j) ;

ξ = the vector of the random variables corresponding to the state of traffic conditions;

Ξ = the support of ξ ; it is assumed to be finite;

t_{ijk}^ξ = the travel time of vehicle k on arc (i, j) if the state is ξ ;

$x_{ijk} = 1$ if the arc (i, j) is traversed by vehicle k and 0 otherwise;

$z_{ik} = 1$ if the node i is visited by vehicle k and 0 otherwise;

$W_k(\xi)$ = the total waiting cost of vehicle k if the state is ξ ;

$S_k(\xi)$ = the total late delivery penalty of vehicle k if the state is ξ ;

$Y_k(\xi)$ = the total overtime cost of vehicle k if the state is ξ ;

$A_{ik}(\xi)$ = the arrival time at vertex i if it is visited by vehicle k ;

$w_{ik}(\xi)$ = the waiting time at vertex i if it is visited by vehicle k and the arrival time is earlier than a_i , and the state is ξ ;

$s_{ik}(\xi)$ = the late delivery time at vertex i if it is visited by vehicle k and the arrival time is later than b_i , and the state is ξ ;

$y_k(\xi)$ = the total overtime time of vehicle k and the state is ξ ;

P_k = the path $(0, i_1, i_2, \dots, i_{d-1}, i_d, n+1)$ travelled by vehicle k ;

Also, define $(i, j) \in P_k$ if i is strictly followed by j in the path P_k and $|P_k|$ is the

length of the path.

The first stage problem becomes then

(P3.1)

$$\min_{x,z} \sum_{k \in K} f_k z_{0k} + \sum_{k \in K} \sum_{(i,j) \in A} c_{ijk} x_{ijk} + E_{\xi} \left(\sum_{k \in K} W_k(\xi) + \sum_{k \in K} S_k(\xi) + \sum_{k \in K} Y_k(\xi) \right) \quad (3.1)$$

subject to

$$\sum_{k \in K} z_{ik} = 1 \quad (i \in V) \quad (3.2)$$

$$\sum_{\substack{j=1 \\ i \neq j}}^n x_{ijk} = z_{ik} \quad (k \in K; i \in V \cup \{0\}) \quad (3.3)$$

$$\sum_{\substack{j=1 \\ i \neq j}}^n x_{jik} = z_{ik} \quad (k \in K; i \in V \cup \{n+1\}) \quad (3.4)$$

$$\sum_{\substack{j=1 \\ i \neq j}}^n x_{ijk} - \sum_{\substack{j=1 \\ i \neq j}}^n x_{jik} = 0 \quad (k \in K; i \in V) \quad (3.5)$$

$$\sum_{k=1}^K D_i z_{ik} \leq C_k \quad (k \in K) \quad (3.6)$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1 \quad (S \subseteq V, 2 \leq |S| \leq n-1; k \in K) \quad (3.7)$$

$$W_k(\xi) - \left(\sum_{i \in P_k} \beta_k^+ w_{ik}(\xi) \right) \left(\sum_{(i,j) \in P_k} x_{ijk} - |P_k| + 2 \right) \geq 0 \quad (\forall P_k; k \in K; \xi \in \Xi) \quad (3.8)$$

$$S_k(\xi) - \left(\sum_{i \in P_k} \beta_i^- s_{ik}(\xi) \right) \left(\sum_{(i,j) \in P_k} x_{ijk} - |P_k| + 2 \right) \geq 0 \quad (\forall P_k; k \in K; \xi \in \Xi) \quad (3.9)$$

$$Y_k(\xi) - \gamma_k y_k(\xi) \left(\sum_{(i,j) \in P_k} x_{ijk} - |P_k| + 2 \right) \geq 0 \quad (\forall P_k; k \in K; \xi \in \Xi) \quad (3.10)$$

$$x_{ijk} \in \{0,1\} \quad (k \in K; (i,j) \in A) \quad (3.11)$$

$$z_{ik} \in \{0,1\} \quad (k \in K; i \in V_0) \quad (3.12)$$

$$W_k(\xi) \geq 0 \quad (k \in K; \xi \in \Xi) \quad (3.13)$$

$$S_k(\xi) \geq 0 \quad (k \in K; \xi \in \Xi) \quad (3.14)$$

$$Y_k(\xi) \geq 0 \quad (k \in K; \xi \in \Xi) \quad (3.15)$$

The objective of this problem is to find a set of tours with minimum sum of the fixed cost, travel cost, and the expected waiting, overtime cost and late delivery penalty. Constraints (3.2) ensure each node is visited by exactly one vehicle. Constraints (3.3) to (3.5) are standard flow balance constraints. Constraints (3.6) are the capacity limit of each vehicle. Constraints (3.7) can eliminate any subtours that are not starting from and ending at the depot. Constraints (3.8) to (3.10), named as the *second-stage constraints*, are gradually added to the problem after solving the second stage problem. They will be discussed in next section. Constraints (3.8) to (3.10) combined with (3.13) to (3.15) reflect the lower bound of the waiting cost, late delivery penalty, and overtime cost and by the route served by vehicle k , which is denoted as P_k .

The second stage problem is a simple recourse model. After solving the first stage problem, a set of routes can be obtained. We name this solution as the *first stage solution*. Then, the stochastic variables can be realized. With a given state ξ and a first-stage solution (x, z) , the second stage problem can be described as follows:

(P3.2)

$$\min_{w, y, s} \sum_{k=1}^K \sum_{i=1}^n (\beta_k^+ w_{ik}(\xi) + \beta_i^- s_{ik}(\xi)) + \sum_{k=1}^K \gamma_k y_k(\xi)$$

subject to

$$A_{jk}(\xi) = A_{ik}(\xi) + w_{ik}(\xi) + \tau_{ik} + t_{ijk}^\xi \quad (k \in K; (i, j) \in P_k) \quad (3.16)$$

$$w_{ik}(\xi) \geq a_i - A_{ik}(\xi) \quad (k \in K; i \in P_k \setminus \{0, n+1\}) \quad (3.17)$$

$$s_{ik}(\xi) \geq A_{ik}(\xi) - b_i \quad (k \in K; i \in P_k \setminus \{0, n+1\}) \quad (3.18)$$

$$y_k(\xi) \geq A_{n+1,k}(\xi) - e_k \quad (k \in K) \quad (3.19)$$

$$A_{0k}(\xi) = d_k \quad (k \in K) \quad (3.20)$$

$$w_{ik}(\xi) \geq 0 \quad (k \in K; i \in V_0) \quad (3.21)$$

$$y_k(\xi) \geq 0 \quad (k \in K) \quad (3.22)$$

$$s_{ik}(\xi) \geq 0 \quad (k \in K; i \in V_0) \quad (3.23)$$

The objective of the second stage problem is to determine the arrival time of each vertex such that the sum total waiting cost, late delivery penalty and overtime cost is minimized. Constraints (3.16) are the computations of the arrival time from one loading point i to another loading point j . Constraints (3.17) to (3.18) together with (3.21) to (3.23) provide the lowest bound of the waiting times, late delivery time and the overtime respectively.

However, computing the optimal second stage solution can be replaced by a simple heuristic approach. For each of the route, we can trace the path starting from depot. Assume every vehicle starting at its earliest working time d_k . Then, for each of the state ξ , the arrival time of each vertex will be the departure time of the previous vertex plus the travel time at that state. If the vehicle arrives earlier than the time window, waiting is required until the start of the service time. Conversely, if the vehicle arrives later than the time window, the vehicle will leave immediately after the service. By using this approach, the second stage problem can be transformed to a simple problem. The approach is outlined as follows:

Algorithm 3.1

Begin

For each ξ do

$$A_{0k}(\xi) := d_k \text{ for all } k \in K$$

$$w_{ik}(\xi) := 0 \text{ for all } k \in K \text{ and all } i \in V_0$$

$$s_{ik}(\xi) := 0 \text{ for all } k \in K \text{ and all } i \in V_0$$

$$y_k(\xi) := 0 \text{ for all } k \in K$$

For each route P_k do

$$P_k := (0, i_1, i_2, \dots, i_d, n+1)$$

For each node in $P_k \setminus \{0, n+1\}$ do

$$A_{i_{p+1}k}(\xi) := A_{i_pk}(\xi) + w_{i_pk}(\xi) + \tau_{i_pk}(\xi) - t_{i_{p+1}k}^\xi$$

If $A_{i_{p+1}k}(\xi) < a_{i_{p+1}}$ then

$$w_{i_pk}(\xi) := a_{i_{p+1}} - A_{i_{p+1}k}(\xi)$$

Else if $A_{i_{p+1}k}(\xi) > b_{i_{p+1}}$ then

$$y_{i_pk}(\xi) := A_{i_{p+1}k}(\xi) - b_{i_{p+1}}$$

End if

End for

$$A_{n+1,k}(\xi) := A_{i_qk}(\xi) + w_{i_qk}(\xi) + \tau_{i_qk}(\xi) + t_{i_qdk}^\xi$$

If $A_{n+1,k}(\xi) > e_k$ then

$$s_k(\xi) := A_{n+1,k}(\xi) - e_k$$

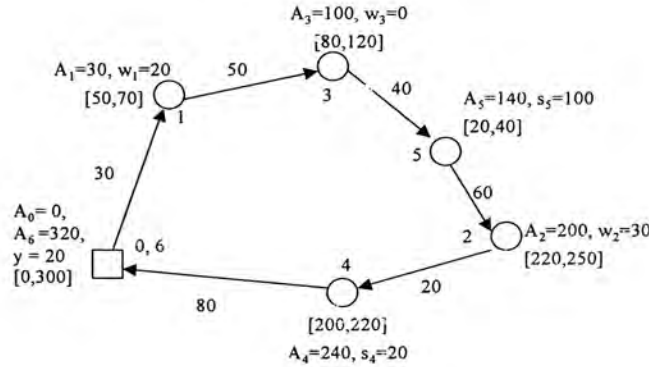
End if

End for

End for

After solving the second stage problem, the waiting cost, late penalty and overtime costs are obtained. Initially, the first stage problem does not include constraints (3.8) to (3.10) and so these costs are not taken into consideration. By imposing these constraints after every second stage problem, we can provide a lower bound of the stochastic costs to a particular first stage solution. Then, these costs can be included in the objective. Example 3.1 will describe how to add these constraints and compute such costs using Algorithm 3.1.

Example 3.1 Consider after solving the first stage problem, the optimal path of a particular vehicle k is $P_k = (0, 1, 3, 5, 4, 2, 6)$. Index 0 and 6 denote the depot. For a particular state ξ , the travel times are realized and listed on the arcs. The time windows are also stated in the following figure.



By using Algorithm 3.1, the arrival time A , the waiting time w , the late delivery time s and the overtime y are computed and stated above. Assume $\alpha_k = \beta_k^+ = \beta_i^- = 1$, then the total waiting cost of this route at state ξ will be equal to 50, the late delivery penalty will be 120 and overtime cost will be 20.

Then, three constraints of type (3.8), (3.9), and (3.10) can be added to the first stage problem like follows:

$$W_k(\xi) - 50(x_{01} + x_{13} + x_{35} + x_{52} + x_{24} + x_{46} - 5) \geq 0 \quad (3.8a)$$

$$S_k(\xi) - 120(x_{01} + x_{13} + x_{35} + x_{52} + x_{24} + x_{46} - 5) \geq 0 \quad (3.9a)$$

$$Y_k(\xi) - 20(x_{01} + x_{13} + x_{35} + x_{52} + x_{24} + x_{46} - 5) \geq 0 \quad (3.10a)$$

From Example 4.1, the path P_k is valid when $x_{01} = x_{13} = x_{35} = x_{52} = x_{24} = x_{46} = 1$. Therefore, when one of these variables equals to 0, the lower bounds of the costs will become 0, as restricted by constraints (3.13) to (3.15). That means, these costs are not taken into the objective value unless this path is chosen. By iteratively solving the first stage and second stage problem, adding constraints (3.8), (3.9) and (3.10), the optimal solution can be obtained. Using this idea, a branch-and-cut algorithm is proposed in next section.

3.3 General Branch-and-cut Algorithm

In this section, we briefly describe a general branch-and-cut algorithm for solving this problem. The algorithm can be outlined as follows:

Algorithm 3.2

Step 0. Initialization: Let z_{opt} be the cost of best known feasible solution. If no solution is known, set $z_{opt} := \infty$. Define the first problem as the relaxed problem containing constraints (3.2) to (3.6). Insert the current problem into a *problem list*.

Step 1. If the problem list is empty, print the best known solution and stop.
Otherwise, select the problem from the list.

Step 2. Solve the current problem and let z_{cur} be the values of its optimal solution. If $z_{cur} > z_{opt}$, fathom the current problem and go to Step 1.

Step 3. If the current solution does not satisfy integrality constraints (3.11) and (3.12), create subproblems by branching on a fractional variable. Insert them into the problem list and go back to Step 1.

Step 4. At an integer solution, check for the existence of subtours. If any subtours exist, introduce appropriate subtour elimination constraints (3.7) to the current problem and go to Step 2.

Step 5. At a first stage solution, let z'_{cur} be the sum of the fixed costs, travel cost, and the expected waiting costs, penalties and overtime costs by using Algorithm 4.1. Introduce suitable constraints (3.8), (3.9) and (3.10) to the current problem. Also, update $z_{opt} = z'_{cur}$ if $z'_{cur} < z_{opt}$ and go to Step 2.

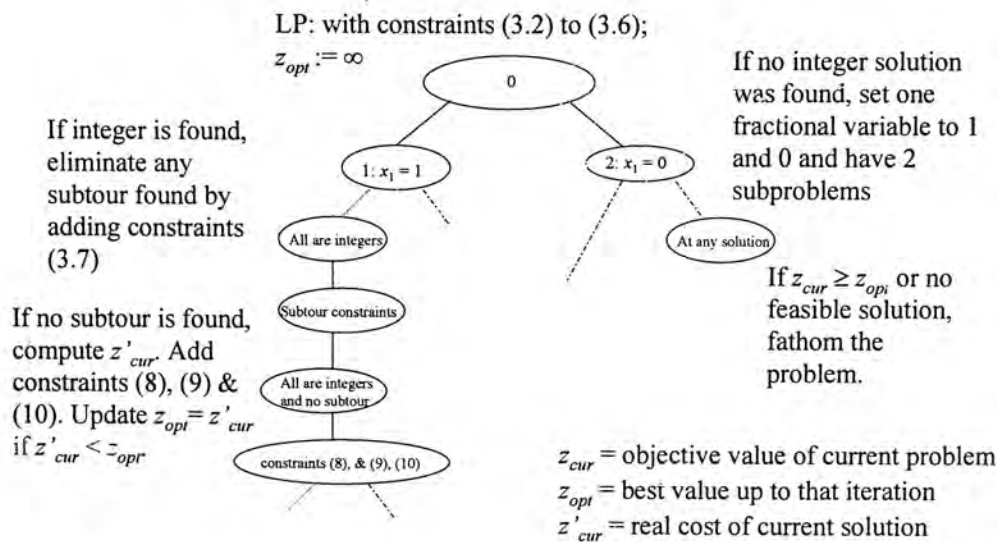


Figure 3.1 Graphical Presentation of Algorithm 3.2

The convergence of this branch-and-cut algorithm is guaranteed by finite number of feasible first stage solutions. Also, the number of second stage constraints is finite. Therefore, this algorithm is exact and with potential $2^{Kn(n+1)+K(n+1)}$ number of pivots in the branching tree. That means, in the worst case, the algorithm needs to examine all the potential pivots, and that would be a very large number, which is not computationally affordable. To avoid this to occur, a better branching and bounding rule may help to find the optimal or good solution earlier so that more branches can be fathomed in earlier iterations.

3.4 Improvement to the Solution Time of the Algorithm

Branch-and-bound algorithm is a typical technique in solving integer programming. By relaxing the integrality constraints of the integer variables and fixing one of them be either 0 or 1 iteratively, a tree can be built and the optimal solution can be eventually obtained. The bound of the tree is the lowest objective value (minimization) of the feasible solutions obtained so far. Any nodes of the tree with objective values higher than the bound can be fathomed away (taken out from consideration), since any solution along that branch cannot be optimal. Because of this reason, the height of the tree will depend on the selection of the branching variables and the bound obtained. Obviously, a tighter bound, if it is obtained earlier, can help to reduce the number of nodes in the tree, hence the time to get the optimal solution. However, in solving stochastic integer programming, a tight bound is very difficult to achieve because the real objective value (include the fixed and stochastic values) cannot be realized until the first stage solution is computed. This restriction leads to a relatively long solution time

since the fathoming process cannot be applied until a tight bound of the real objective value is obtained. Therefore, in this section, we present some methods to, firstly prefix some variables (Pre-fixing), secondly introduce more second-stage constraints than that in Section 3.2 (Directed Partial Path Inequalities), and lastly impose the expected waiting and penalty costs to the coefficients of the flow variables in the objective function (Exponential Smoothing), in order to reduce the solution time of the branch-and-cut algorithm.

3.4.1 Pre-fixing

Pre-fixing variable herein means to set some of the flow decision variables x_{ijk} equal to 0, if they are unlikely equal to 1 in the optimal solution. This process can reduce the number of variables taken into consideration. As the solution time of the branch-and-cut algorithm is exponentially increasing to the number of the integer variables, reduction of the number of variables can assist to decrease the solution time obviously.

In order to determine the unlikelihood of the flow decision variables equaling to 1 in the optimal solution, a minimum expected waiting-and-penalty cost matrix, Φ_k , is established to compute the minimum expected waiting cost or late penalty to each flow decision from node i to node j by vehicle k . Then we have the following definitions.

Definition 3.1 *Let two nodes be i and j , $\forall i, j \in V$, with their time windows $[a_i, b_i]$ and $[a_j, b_j]$ respectively. Define the minimum waiting time from node i to node j by vehicle k at state ξ be $\omega_{ijk}(\xi)$ where*

$$\omega_{ijk}(\xi) = \max\{a_j - (b_i + t_{ijk}^\xi), 0\}.$$

Definition 3.2 Let two nodes be i and j , $\forall i, j \in V$, with their time windows $[a_i, b_i]$ and $[a_j, b_j]$ respectively. Define the minimum late arrival time from node i to node j by vehicle k at state ξ be $\psi_{ijk}(\xi)$ where

$$\psi_{ijk}(\xi) = \max\{(a_i + t_{ijk}^\xi) - b_j, 0\}$$

Definition 3.3 Define ϕ_{ijk} be the minimum expected waiting costs and late penalty from node i to node j by vehicle k where

$$\phi_{ijk} = E_\xi(\beta_k^+ \omega_{ijk}(\xi) + \beta_i^- \psi_{ijk}(\xi))$$

and $\Phi_k = (\phi_{ijk})$.

It is easy to prove that for a pair of i and j of any vehicle k at any state ξ , either $\omega_{ijk}(\xi) > 0$ or $\psi_{ijk}(\xi) > 0$ but not both. Therefore, it is impossible to double count the minimum waiting cost $\beta_k^+ \omega_{ijk}(\xi)$ and the minimum late penalty $\beta_i^- \psi_{ijk}(\xi)$ in ϕ_{ijk} . After establishing the matrix Φ_k for each vehicle k , we can set $x_{ijk} = 0$ whenever ϕ_{ijk} is greater than a given threshold Δ . One meaning of the value of Δ is to demonstrate the acceptance level of the decision maker to the deviation of the delivery to the time window. The larger of the value of Δ , the more accurate of the algorithm is.

Another physical meaning of the pre-fixing technique is to reduce the chance of having unreasonable first stage solutions. Since the initial first stage problem does not consider any delivery time windows, the returned optimal solution will merely minimize the fixed and travel costs, which usually violates

the sequence of the time windows seriously. Therefore, by applying this technique, we can avoid to obtain first stage solution with late order followed by early order (i.e. $a_i \gg b_j$ and $x_{ijk} = 1$) since ϕ_{ijk} is already greater than Δ in this case. Hence, a better solution with lower bound can be obtained in earlier iterations.

The effectiveness of the pre-fixing technique depends on the width of the time window. If the time window is generally wide for majority of the nodes, the possibility of fulfilling the time windows will be larger, that is $\phi_{ijk} = 0$ for more i, j and k . Therefore, less number of variables are fixed in this case. The situation is completely reverse if the length of the time window is shorter in general. However, a problem with longer time window is “easier” to solve in the sense that less second stage constraints are generated and so the number of iterations in resolving the first stage problem is reduced. Thus, pre-fixing technique is useful when the length of the time window is relatively short comparing with the length of the planning horizon.

3.4.2 Directed Partial Path Valid Inequalities

The formulation of the VRPTWST can achieve optimality because the first stage problem is iteratively resolved as the expected waiting/late cost of each route (second stage constraints) is gradually included. The difficulty in solving the VRPTWST with soft time window is that the waiting/late cost cannot be realized until a complete route in a particular sequence is obtained. As a result, for each of the first stage solution, we can only add second stage constraints for a set of routes and each route is in a specific sequence. In this section, we will introduce

the idea of adding more second stage constraints other than the original set of constraints.

Recalling Example 4.1, the optimal first stage solution is the route $P^1 = (0, 1, 3, 5, 2, 4, 6)$ with waiting cost = 50, late penalty = 120 and overtime cost = 20. Suppose the fixed and travel cost of this route is 300. The total cost of this route will be 490. Then, constraints (3.8a) to (3.10a) will be added to the first stage problem. After resolving the augmented first stage problem, another optimal solution $P^2 = (0, 1, 3, 5, 4, 2, 6)$ is obtained with fixed and travel cost equal to, say 320. Obviously, the optimal solution of the augmented problem will no longer be P^1 because the objective value of solution P^1 will be 490. Figure 3.2 shows the difference between the solution paths of P^1 and P^2 .

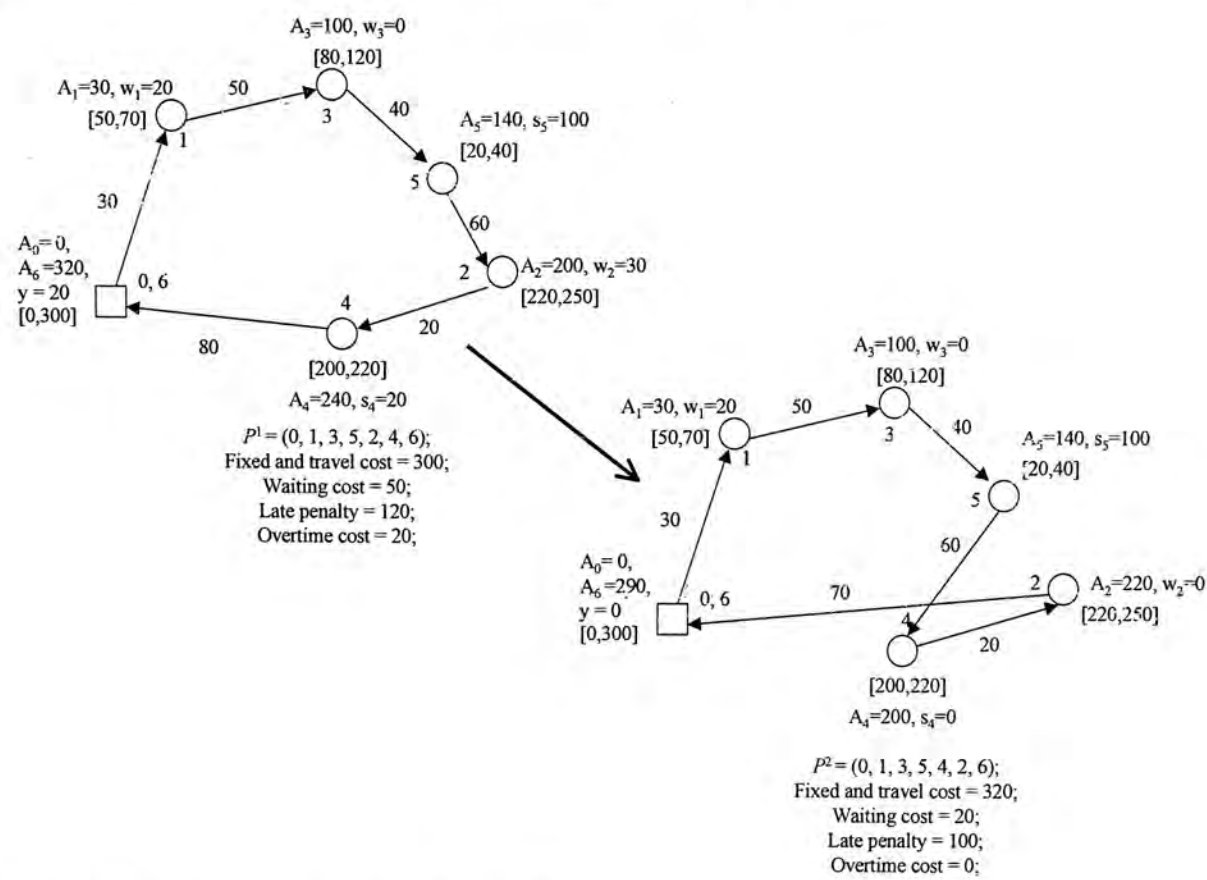


Figure 3.2 Solution paths for P^1 and P^2

The difference between P^1 and P^2 in term of their sequences is only the

visiting order of the last two nodes, which are node 2 and node 4. From the values listed in the Figure 3.2, the total cost of P^2 is 440 that show an improvement. However, if we trace the path again and we can find that the most serious penalty occurs at node 5, which is at the middle (beginning) of the path. Suppose, in a larger case, there are more nodes to be visited after node 5. Adding merely constraints in type (3.8) and (3.9) can only include the waiting/late cost of *one-and-only-one* sequence of a route in each iteration. And hence, the first stage solutions in the following iterations may only swap the visiting orders of the nodes at the tail of the route. This circumstance may need a huge amount of iterations to come up with a solution so that all nodes' time windows are well satisfied. Although this situation may not happen frequently, it cannot be completely avoided. Thus, we introduce more second stage constraints to relieve the problem.

One of the natures of the solution path is that the accumulated waiting/late cost up to a node does not depend on any subsequent nodes. That means, in Example 4.1, the accumulated late penalty after visiting node 5 is 100. Using Algorithm 3.2, if a path follows the sequence (0, 1, 3, 5), the late penalty must be 100 after visiting node 5 no matter what is the path after that. This property also applies to the accumulated waiting cost. Using this idea, we can add several more second stage constraints like the following:

$$W_k(\xi) - 20(x_{01} + x_{13} - 1) \geq 0$$

$$W_k(\xi) - 20(x_{01} + x_{13} + x_{35} - 2) \geq 0$$

$$W_k(\xi) - 50(x_{01} + x_{13} + x_{35} + x_{52} - 3) \geq 0$$

$$W_k(\xi) - 50(x_{01} + x_{13} + x_{35} + x_{52} + x_{24} - 4) \geq 0$$

$$S_k(\xi) - 100(x_{01} + x_{13} + x_{35} - 2) \geq 0$$

$$S_k(\xi) - 100(x_{01} + x_{13} + x_{35} + x_{52} - 3) \geq 0$$

$$S_k(\xi) - 120(x_{01} + x_{13} + x_{35} + x_{52} + x_{24} - 4) \geq 0$$

Instead of adding the second stage constraints of waiting/late costs of the entire path, the path now is decomposed in to many sub-paths and each of them start from the depot, follow the original path and end at one of the node. Therefore, these constraints are named as *Directed Partial Path Valid Inequalities* (DPPVI).

Define S_k^v be a sub-path of P_k where the length of the sub-path is v , $0 \leq v \leq |P_k|-1$.

Also, $S_k^0 = (0)$ and $S_k^{|P_k|-1} = P_k$. For Example 4.1, $S_k^2 = (0,1,3)$ and $S_k^3 = (0,1,3,5)$.

Then, constraints (3.8) and (3.9) can be modified to the following general forms.

$$W_k(\xi) - \left(\sum_{i \in S_k^v} \beta_k^+ w_{ik}(\xi) \right) \left(\sum_{(i,j) \in S_k^v} x_{ijk} - |S_k^v| + 2 \right) \geq 0$$

$$(1 < v \leq |P_k| - 1; k \in K; \xi \in \Xi) \quad (3.8b)$$

$$Y_k(\xi) - \left(\sum_{i \in S_k^v} \beta_i^- w_{ik}(\xi) \right) \left(\sum_{(i,j) \in S_k^v} x_{ijk} - |S_k^v| + 2 \right) \geq 0$$

$$(1 < v \leq |P_k| - 1; k \in K; \xi \in \Xi) \quad (3.9b)$$

By imposing the new set of constraints (3.8b) and (3.9b), the number of first stage solution is reduced. Its efficiency will be showed in Section 4.6. This technique is efficient in reducing the solution time when the fixed/travel cost contributes a small amount (relative to the waiting/late cost) to the total cost.

3.4.3 Exponential Smoothing

In the formulation of the SVRP presented in Section 3.2, the objective function of the first stage problem includes three items, which are fixed costs, travel costs and expected waiting/late/overtime costs. At the earlier iterations of the

branch-and-cut procedure, value of the last item equals to zero since only a few of second stage constraints are included (i.e. expected costs of a few number of routes are considered). Therefore, the first stage problem is trying to minimize the fixed and travel costs only (a type of geographical optimization) without considering any costs related to the time windows in the beginning iterations. Consequently, the first stage solutions obtained at the beginning usually violate the time windows of the nodes. The objective value of the early first stage solution generally has a low value of fixed/travel cost but extremely high value of waiting/late cost. This is again because the routes obtained do not follow the time window requirements. Due to this reason, the initial bounds from this formulation will not be tight,

so slowing down the fathoming process. In this section, we try to modify the objective function so that partial value of the expected waiting/late cost is included in the coefficient of each flow variable. Since the expected waiting/late cost will be reflected in the last term of the objective function, this partial value must be reduced during the iterations so that this cost will not be double counted. This partial value is exponentially diminished during the solving iterations. Therefore, this technique is named as *Exponential Smoothing*. In the following paragraphs, we will discuss how to determine this partial value and how does the technique work.

The first amendment of this modification is to add some value of the expected waiting/late cost to the coefficient (i.e. c_{ijk} originally) of the flow variable (x_{ijk}). In the original formulation, c_{ijk} is the travel cost from node i to node j by vehicle k and it is not time-dependent. That is, if we select to use vehicle k traveling from i to j , a cost c_{ijk} must be incurred no matter when does

the vehicle leaves node i . However, it is not able to identify an exact value of the expected waiting/late cost adding to coefficient of x_{ijk} , like c_{ijk} , since such cost depends on whether if vehicle k travel from i to j can arrive within the delivery time window of node j . For this reason, we can only add the expected minimum waiting/late cost ϕ_{ijk} , defined in Definition 3.3, to the coefficient. Because ϕ_{ijk} is the expected minimum cost that must be incurred if vehicle k travels from node i to node j , adding this value to the coefficient will not overestimate the objective value. The objective function of the first stage becomes

$$\min_{x,z} \sum_{k \in K} f_k z_{0k} + \sum_{k \in K} \sum_{(i,j) \in A} (c_{ijk} + \phi_{ijk}) x_{ijk} + E_{\xi} \left(\sum_{k \in K} W_k(\xi) + \sum_{k \in K} S_k(\xi) + \sum_{k \in K} Y_k(\xi) \right) \quad (3.1a)$$

After the value ϕ_{ijk} is added to the coefficient of x_{ijk} , as the iterations go on, there is a chance to over count the expected waiting/late cost. The insertion of the second stage constraints will completely reflect the waiting/late cost of a route to the objective value. If the value ϕ_{ijk} still exists in the coefficient of x_{ijk} , the objective value will be exaggerated for that route. This may affect the fathoming process as the real objective value may be lower. Therefore, the contribution of ϕ_{ijk} to the objective function must be gradually decreased during the solving iterations process.

In order to reduce the value ϕ_{ijk} during the iterations, a multiplier α_{ijk} with updating is multiplied to ϕ_{ijk} . Given a parameter λ , $0 < \lambda < 1$, a updating scheme is outlined in Algorithm 3.3 as follow:

Algorithm 3.3

Begin

Given $0 < \lambda < 1$;

Initialize $\alpha_{ijk} := 1$;

For each first stage solution found **do**

If $x_{ijk} = 1$ in the first stage solution **then**

$$\alpha_{ijk} := \alpha_{ijk} \times \lambda;$$

End if

End for

The updating scheme stated in Algorithm 3.3 could reduce α_{ijk} by a factor of λ each time if x_{ijk} is found to be one in the first stage solution. This scheme is logic in the sense that the expected waiting/late cost of x_{ijk} (a route with $x_{ijk} = 1$) is reflected by the corresponding second stage constraints. By the way, α_{ijk} can be decreased by a portion to reduce the chance of over counting. The choice of value of λ will affect the quality of the solution. If λ is close to one, the chance of double counting is larger and may lead to a solution with greater objective value. On the contrary, if λ is chosen to be small, α_{ijk} will approach zero in few iterations such that the performance tends to be the same as the original formulation.

As the value of ϕ_{ijk} is gradually decreased during the iterations, over counting the expected waiting/late cost cannot be completely avoided. Therefore, applying this method will no longer guarantee the optimality of the

branch-and-cut algorithm. However, in term of solution time, it generates a great improvement to the original algorithm that will be demonstrated in Section 4.6.

3.4.4 Fast Fathoming

In this section, we present one more technique to reduce the solution time of the algorithm. The idea of this technique is to apply the fathom process earlier than the original algorithm while guaranteeing the difference between the real optimal value and the optimal value using this technique is bounded by a given percentage. This technique is referred as *Fast Fathoming* and must be applied together with the Exponential Smoothing technique.

In typical branch-and-bound process of a minimization problem, fathoming occurs when the objective value of the current subproblem, z_{cur} , is greater than the best value, z_{opt} , obtained so far. (i.e. $z_{cur} > z_{opt}$). If fathoming is now applied to any subproblems with $(1+\mu) z_{cur} > z_{opt}$, where μ is a given value between 0 and 1, we have the following lemma.

Lemma 3.1 *During a branch-and-bound process of a minimization problem, if fathoming is applied to any subproblem with $(1+\mu) z_{cur} > z_{opt}$, where z is the objective value of the subproblem, z^* is the best objective value obtained so far, and μ is a given value between 0 and 1, then the optimal value under this fathoming rule, denoted as Z_μ , must be smaller than $(1+\mu) Z_0$, where Z_0 is the optimal value solution of fathoming rule with $\mu = 0$.*

Proof: Suppose z^* is the best objective value obtained so far. A node with objective value z will be fathomed away if $(1+\mu) z_{cur} > z_{opt}$. Since z_{cur} is the lower

bound value of the subtree under that node, all nodes of the subtree must have objective values greater than or equal to z_{cur} . Let the real optimal solution is one of the nodes in this subtree, with objective value Z_0 , and suppose the best solution value obtained under fast fathoming is $Z_\mu = z_{opt}$. Therefore, we have,

$$Z_0 \geq z_{cur}, \text{ or } (1 + \mu)Z_0 \geq (1 + \mu)z_{cur}$$

and by definition, $(1 + \mu)z_{cur} \geq z_{opt}$

$$(1 + \mu)Z_0 \geq z_{opt} = Z_\mu \quad \square$$

The fast fathoming technique can be applied to our branch-and-cut algorithm. It would be more effective, if the value ϕ_{ijk} is added to the objective function, because the objective values of solving these subproblems are closer to the real total costs. Since the objective function of the problem only reflects the fixed/travel costs and minimum expected waiting/late costs, the real cost of the solution is usually larger than the objective value (computed directly from the objective function). Thus, the difference between the real optimal value, Z_0 , and the optimal value using this technique, Z_μ , is usually much smaller than theoretical bound (i.e. $Z_0 \leq Z_\mu \ll (1 + \mu)Z_0$). This special property will be demonstrated in the numerical results in Section 3.5.

In our analysis, we try to apply this fast fathoming technique at two different stages. Recalling Algorithm 4.1, the first scenario is to apply fast fathoming with μ_1 at Step 1 of the algorithm, which is referred as the *First-stage Fathoming*, $FF(\mu_1)$. The second scenario is to apply fast fathoming with μ_2 at Step 5 if $(1 + \mu_2)z_{cur} > z_{opt}$, which is referred as *Second-stage Fathoming*, $SF(\mu_2)$. Generally, μ_2 is chosen to be larger than μ_1 because the Second-stage

Fathoming affects less number of nodes. In addition, fast fathoming can be used at both the first stage and the second stage with parameter μ_1 and μ_2 respectively. This is referred as *Both-stage Fathoming*, $\text{BF}(\mu_1, \mu_2)$.

3.4.5 Modified Branch-and-cut Algorithm

In the previous sections, we introduce several techniques to improve the solution time of the original branch-and-cut algorithm. These techniques include Pre-fixing, Directed Partial Path Valid Inequalities, Exponential Smoothing and Fast Fathoming. After imposing these techniques to the branch-and-cut algorithm, the solution to the problem becomes a heuristic solution. In this section, the modified branch-and-cut algorithm with all 4 techniques included is outlined as follows.

Algorithm 3.4

Step 0. Initialization: Given the parameters Δ , λ , μ_1 and μ_2 . Set $\alpha_{ijk} := 1$ and find the value ϕ_{ijk} for all i, j and k . Set $x_{ijk} := 0$ if $\phi_{ijk} > \Delta$. Let z_{opt} be the cost of best known feasible solution. If no solution is known, set $z_{opt} := \infty$. Define the first problem as the relaxed problem containing objective function (3.1a) and constraints (3.2) to (3.6). Insert the current problem into a problem list.

Step 1. If the problem list is empty, print the best known solution and stop. Otherwise, select the problem from the list.

Step 2. Solve the current problem and let z_{cur} be the value of its optimal solution. If $(1 + \mu_1)z_{cur} > z_{opt}$, fathom the current problem and go to Step 1.

Step 3. If the current solution does not satisfy integrality constraints (3.11) and (3.12), create subproblems by branching on a fractional variable. Insert them into the problem list and go back to Step 1.

Step 4. At an integer solution, check for the existence of subtours. If any subtours exist, introduce appropriate subtour elimination constraints (3.7) to the current problem and go to Step 2.

Step 5. At a first stage solution, let z'_{cur} be the sum of the fixed costs, travel cost, and the expected waiting costs, penalties and overtime costs by using Algorithm 4.1. If $z'_{cur} > z_{opt}$, fathom the current solution only if $(1 + \mu_2) z_{cur} > z_{opt}$ and go to Step 1. Otherwise, introduce suitable constraints (3.8b), (3.9b) and (3.10) to the current problem. Update $\alpha_{ijk} := \alpha_{ijk} \times \lambda$ according to Algorithm 3.3. Also, update $z_{opt} = z'_{cur}$ if $z'_{cur} < z_{opt}$ and go to Step 2.

3.5 Computational Analysis

In this section, we are going to analyze the effect of each technique to the solution time and optimality. The analysis will be carried into two stages. The first stage will compare the effectiveness of each Pre-fixing, Decomposing and Exponential Smoothing technique on relatively small size problems. After the confirmation of the success of each technique, we will compare the performance of the algorithm with and without Fast Fathoming on larger size problems in the second stage analysis.

3.5.1 Performance of Pre-fixing, Directed Partial Path Valid Inequalities and Exponential Smoothing

In this part, comparisons are made between the performance of original branch-and-cut algorithm and that when each technique, Pre-fixing, Decomposing and Exponential Smoothing, is added to the algorithm. Two scenarios: 1) $n = 6$, $k = 2$ and $|\Xi| = 3$, and 2) $n = 10$, $k = 2$ and $|\Xi| = 3$, are investigated. The statistics of 50 samples of the following aspects will be reported:

- Optimal value
- Solution CPU time
- Number of iterations
- Number of Pre-fixing variables (if any)
- Number of nodes in branching-and-bound tree
- Number of optimality cuts
- Number of first stage solutions

(1) $n = 6$, $k = 2$ and $|\Xi| = 3$

In the first scenario, there are 6 nodes to be served and 2 vehicles located at a single depot. The locations of the 6 vertices are uniformly distributed in an area of $20 \text{ km} \times 20 \text{ km}$ and the depot is located at the center of the area. There are 3 traffic states, namely normal, congested and very congested, and each state has different impact on the travel speed in the area. Distance between nodes are assumed to be Euclidean. The length of the planning horizon is 4 hours. All parameters of the problem are summarized in Table 3.1.

Parameters	Values
<u>Planning Horizon</u>	0000-0240 (i.e. 4 hours)
<u>Area</u>	20 km × 20 km
<u>Nodes</u>	
- Demand, D_i	Normal (800, 100)
- Start of Time Window, a_i	Uniform (0000, 0240)
- Length of Time Window, $(b_i - a_i)$	Uniform (0.5hr, 1.0hr)
- Late Penalty per minute, β_i^-	$0.5 \times D_i / \text{Mean Demand}$
<u>Vehicles</u>	
- Capacity, C_k	Normal (3000, 1000)
- Waiting Cost per minute, β_k^+	0.05
<u>Traffic States</u>	
- $Prob(\text{Normal}) = 0.5$	Speed: Uniform (45km/hr, 50km/hr)
- $Prob(\text{Congested}) = 0.4$	Speed: Uniform (35km/hr, 45km/hr)
- $Prob(\text{Very Congested}) = 0.1$	Speed: Uniform (30km/hr, 35km/hr)

Table 3.1 Summary of the parameters of the sample problem when $n = 6$, $k = 2$ and $|\Xi| = 3$

From the average values of the 100 sample runs, we can conclude that each of the 3 techniques can help to reduce the solution time of the algorithm. In terms of optimality, Pre-fixing technique obviously depends on the size of the threshold Δ . In this analysis, we choose $\Delta = 400$ and it can fix 14.87 variables on average, which is equivalent to 15.7% of binary variables. The average increase of the objective value is only 0.0 to 2.0 % but helping to reduce solution time by nearly 50% on average.

For adding Directed Partial Path Valid Inequalities, since it only add more constraints to reflect the waiting/late cost of more routes only. Therefore, it does not change the objective value (as expected). In terms of solution time, it only reduces relatively small amount comparing with other 2 techniques. This may be due to the reason that the linear program becomes more complicated to solve

Average Value of Sample Size, $N=100$	Original ($T=O$)	Pre-fixing ($T=P$) $\Delta=400$	DPPVI ($T=D$)	Exponential Smoothing ($T=E$) $\lambda=0.7$	($T=PDE$) $\Delta=400$ $\lambda=0.7$
Optimal Value (Z_T)	2870.57	2906.94	2870.57	2877.44	2913.69
CPU Time (sec)	260.38	144.85	208.12	40.66	30.51
No. of Iterations	2395.05	1217.73	1566.00	360.82	213.76
No. of Pre-fixing Var(s)	-	14.86	-	-	14.86
No. of B&B Nodes	1658.43	850.46	940.57	298.58	156.49
No. of Optimality Cut	624.89	304.97	419.95	91.29	53.85
No. of 1 st Stage Solution	435.88	222.87	382.45	47.23	35.89

* average utilization of vehicles = 0.74

Table 3.2 Summary of the average values of statistics of the sample problem when $n = 6$, $k = 2$ and $|\Xi| = 3$

Difference to the Original Algorithm of Sample Size, $N=100$	Pre-fixing ($T=P$) $\Delta=400$	DPPVI ($T=D$)	Exponential Smoothing ($T=E$) $\lambda=0.7$	($T=PDE$) $\Delta=400$ $\lambda=0.7$
<u>Optimal Value (Z_T)</u>				
- ^a $\bar{X}(N), S(N)$	0.01, 0.04	0.00, 0.00	0.00, 0.03	0.01, 0.05
- ^b 95% Confidence Interval	± 0.01	± 0.00	± 0.00	± 0.01
- ^c % of optimal value obtained	0.78	1.00	0.87	0.78
<u>Solution Time (sec)</u>				
- $\bar{X}(N), S(N)$	-0.48, 0.32	-0.10, 0.47	-0.82, 0.12	-0.86, 0.10
- 95% Confidence Interval	± 0.05	± 0.08	± 0.02	± 0.02
<u>No. of Iterations</u>				
- $\bar{X}(N), S(N)$	-0.52, 0.28	-0.25, 0.35	-0.83, 0.11	-0.89, 0.07
<u>No. of B&B Nodes</u>				
- $\bar{X}(N), S(N)$	-0.51, 0.31	-0.34, 0.34	-0.80, 0.14	-0.88, 0.08
<u>No. of 1st Solution</u>				
- $\bar{X}(N), S(N)$	-0.53, 0.30	-0.02, 0.39	-0.89, 0.10	-0.91, 0.08

^a $\bar{X}(N) = E\left(\frac{Z_{T=P,D,E} - Z_{T=O}}{Z_{T=O}}\right), S(N) = STDEV\left(\frac{Z_{T=P,D,E} - Z_{T=O}}{Z_{T=O}}\right)$

^b $\bar{X}(N) \pm t_{N-1, 0.95} \sqrt{\frac{S^2(N)}{N}}$

^c percentage of samples that find the same optimal value of original algorithm

Table 3.3 Summary of the difference between the original algorithm and the enhanced algorithms with different techniques added when $n = 6$, $k = 2$, $|\Xi| = 3$

after all directed partial path constraints are added. But in terms of the number of nodes of the branching tree, it does show an improvement. Exponential Smoothing is the most effective technique among the three. With $\lambda = 0.7$, we observe that the solution time decreases dramatically by 80% on average with a resultant increases of the average objective value by less than 0.2%.

If we use all the techniques at the same time, the average solution time drops to 30.51 seconds from 260.38 seconds when no technique is used. The average decrease is about 88%. The objective value only changes by 2% on average, which can be mainly explained the Pre-fixing technique. All related figures are summarized in Table 3.2 and Table 3.3.

Up to this point of time, we are quite sure of the effectiveness of the 3 techniques. Because the problem size in this scenario is still small, we try to do one more scenario with larger size to support our assertion.

(2) $n = 10$, $k = 2$ and $|\Xi| = 3$

This scenario is similar to the first scenario. In this case, there are 10 nodes to be served. To satisfy the capacity of the vehicles, the mean of the demand is adjusted to 450 while keeping the capacity of vehicles constant. Other parameters are the same as stated in Table 4.1. Also, solving this problem size already takes very long time using the original branch-and-cut algorithm. For this reason, the solving process will terminate either up to 80,000 iterations or optimal solution is obtained. The sample size is 50 in this analysis.

In this scenario, since the number of iterations is fixed, optimal values of

Average Value of Sample Size, $N = 50$	Original ($T=O$)	Pre-fixing ($T=P$) $\Delta = 400$	DPPVI ($T=D$)	Exponential Smoothing ($T=E$) $\lambda = 0.7$	($T=PDE$)
Optimal Value (Z_T)	3450.34	3276.53	3784.19	3223.82	3221.57
^a % of optimal solution obtained	20%	53%	40%	93%	100%
Solution Time (sec)	7668.60	5683.20	12512.80	2095.00	932.80
No. of Iterations	76958.53	53468.27	74415.67	14715.47	5426.87
No. of Pre-fixing Var(s)	-	42.58	-	-	42.58
No. of B&B Nodes	52597.53	34528.13	48024.13	11299.40	3849.40
No. of Optimality Cut	15754.27	15463.93	20264.33	4673.20	1589.73
No. of 1st Stage Solution	11321.67	7456.93	13288.40	1611.40	844.60

^a percentage of samples that can terminate within 80,000 iterations

* average utilization of vehicles = 0.72

Table 3.4 Summary of the average values of statistics of the sample problem when $n = 10$, $k = 2$ and $|\Xi| = 3$

Difference to the Original Algorithm of Sample Size, $N = 50$	Pre-fixing ($T=P$) $\Delta = 400$	DPPVI ($T=D$)	Exponential Smoothing ($T=E$) $\lambda = 0.7$	($T=PDE$)
<u>Optimal Value (Z_T)</u>				
- $\bar{X}(N), S(N)$	-0.05, 0.05	0.09, 0.46	-0.06, 0.05	-0.06, 0.05
- 95% Confidence Interval	± 0.021	± 0.20	± 0.021	± 0.21
<u>Solution CPU Time (sec)</u>				
- $\bar{X}(N), S(N)$	-0.26, 0.39	0.65, 0.37	-0.73, 0.45	-0.88, 0.22
- 95% Confidence Interval	± 0.09	± 0.09	± 0.10	± 0.05
<u>No. of Iterations</u>				
- $\bar{X}(N), S(N)$	-0.31, 0.34	-0.02, 0.20	-0.81, 0.30	-0.93, 0.11
<u>No. of B&B Nodes</u>				
- $\bar{X}(N), S(N)$	-0.35, 0.36	-0.09, 0.18	-0.79, 0.34	-0.93, 0.12
<u>No. of 1st Solution</u>				
- $\bar{X}(N), S(N)$	-0.35, 0.34	0.19, 0.27	-0.86, 0.26	-0.86, 0.26

Table 3.5 Summary of the difference between the original algorithm and the enhanced algorithms with different techniques added when $n = 10$, $k = 2$ and $|\Xi| = 3$

a majority of samples cannot be obtained by the original algorithm or algorithm with only one technique. But from the number of samples that can achieve optimality within limited iterations, we can conclude that the 3 techniques can help to reduce the solution time while maintaining near optimality. Also, we find that Exponential Smoothing is the most effective technique and Pre-fixing follows.

3.5.2 Performance of Fast Fathoming

After investigating the performance of the first 3 techniques in a smaller size of problem, we are sure that they can help to reduce the solution time significantly while maintaining a good solution. In this part, we will show the effectiveness of the Fast Fathoming technique. As mentioned in previous, Fast Fathoming technique must be applied together with Exponential Smoothing technique. Therefore, the comparison will be made between the algorithm with PDE only and the algorithm with PDE and Fast Fathoming. Different fathoming techniques including the Fast-stage Fathoming, Second-stage Fathoming and Both-stages Fathoming will be demonstrated. We will consider 2 scenarios: 1) $n = 10$, $k = 2$ and $|\Xi| = 3$ and 2) $n = 15$, $k = 2$ and $|\Xi| = 3$.

(1) $n = 10$, $k = 2$ and $|\Xi| = 3$

In this scenario, the parameters setting are the same as the previous scenario. 3 fast fathoming techniques are examined:

- First-stage fathoming with $\mu_1 = 0.10$, denoted as FF(0.1)
- Second-stage fathoming with $\mu_1 = 0.10$, denoted as SF(0.1)
- Both-stages fathoming with $\mu_1 = 0.05$ and $\mu_2 = 0.10$, denoted as BF(0.05,

0.1)

Average Value of Sample Size, $N=50$	($T=PDE$)	First-stage Fathoming $T=PDE+FF(0.10)$	Second-stage Fathoming $T=PDE+SF(0.10)$	Both-stages Fathoming $T=PDE+BF(0.05,0.10)$
Optimal Value (Z_T)	3589.51	3618.57	3590.43	3606.63
Solution CPU Time (sec)	1736.14	887.00	1219.98	1168.66
No. of Iterations	9322.29	5043.64	6529.94	5757.20
No. of Pre-fixing Variables	42.58	42.58	42.58	42.58
No. of B&B Nodes	6239.22	3214.16	4497.76	3884.72
No. of Optimality Cut	2336.47	1032.22	973.78	1032.22
No. of 1st Stage Solution	1575.84	851.71	1219.30	1001.22

* average utilization of vehicles = 0.72

Table 3.6 Summary of the average values of statistics of the sample problem when $n = 10$, $k = 2$ and $|\Xi| = 3$

Difference to the Original Algorithm of Sample Size, $N = 50$	First-stage Cut $T=PDE+FC(0.10)$	Second-stage Cut $T=PDE+SC(0.10)$	Both-stages Cut $T=PDE+BC(0.05,0.10)$
<u>Optimal Value (Z_T)</u>			
- $\bar{X}(N)$, $S(N)$	0.009, 0.014	0.000, 0.001	0.005, 0.012
- 95% Confidence Interval	± 0.003	± 0.000	± 0.003
- % of solutions equal to the solutions obtained by PDE	56%	96%	68%
- Max difference to PDE	0.058	0.006	0.054
<u>Solution Time (sec)</u>			
- $\bar{X}(N)$, $S(N)$	-0.66, 0.16	-0.30, 0.18	-0.46, 0.22
- 95% Confidence Interval	± 0.04	± 0.04	± 0.05
<u>No. of Iterations</u>			
- $\bar{X}(N)$, $S(N)$	-0.48, 0.81	-0.29, 0.13	-0.47, 0.16
<u>No. of Nodes</u>			
- $\bar{X}(N)$, $S(N)$	-0.59, 0.17	-0.25, 0.15	-0.44, 0.14
<u>No. of 1st Solution</u>			
- $\bar{X}(N)$, $S(N)$	-0.74, 0.16	-0.22, 0.10	-0.55, 0.16

Table 3.7 Summary of the difference between the PDE algorithm and enhanced algorithms with different fast fathoming techniques added when $n = 10$, $k = 2$ and $|\Xi| = 3$

From the figures obtained, we find that First-stage fathoming with PDE reduces the time most significantly. On average, it can reduce 66% of solution time compared with the algorithm with PDE only. Since First-stage fathoming is applied at the earlier stage of the iterative process, the quality of solution obtained is poorer than the other two fathoming techniques. The optimal value obtained is increased by 0.9% on average compared with the algorithm with PDE only.

The second-stage fathoming maintains good optimality since it is applied later in the solution iterations. Only 4% of the samples are different from the optimal value of PDE algorithm and these generate insignificant difference on the average optimal value. It can reduce 30% of solution time on average. The result obtained by both-stages fathoming is in between FF and SF. The optimal value is increased by 0.5% on average and the solution time is reduced by 46%. It can be explained by the fact that the first fathoming parameter ($\mu_1=0.05$) is only half of that in the first-stage fathoming ($\mu_1=0.10$). Average values and comparisons are summarized in Table 3.8 and Table 3.9.

(2) $n = 15$, $k = 2$ and $|\Xi| = 3$

In this scenario, the number of delivery points is increased to 15. The average demand of each point is adjusted to 300 units. The length of the planning horizon is increased to 21,600 seconds (6 hours). Since the algorithm takes quite long time for some samples to finish, the solution process will terminate up to 240,000 iterations or optimal solution is obtained. The results obtained in this scenario agree with the conclusion drawn in the last section. Because First-stage

fathoming is applied at the early step of the iterative process, there will be a

Average Value of Sample Size, $N=50$	($T=PDE$)	First-stage Fathoming $T=PDE+FF(0.10)$	Second-stage Fathoming $T=PDE+SF(0.10)$	Both-stages Fathoming $T=PDE+BF(0.05,0.10)$
Optimal Value (Z_T)	4524.38	4548.38	4529.69	4532.56
Solution Time (sec)	9004.62	2864.36	5456.72	3857.64
No. of Iterations	49514.74	14824.77	30239.77	21147.18
No. of Pre-fixing Variables	130.05	130.05	130.05	130.05
No. of B&B Nodes	36447.00	9939.82	22844.49	15419.92
No. of Optimality Cut	14916.31	486.05	5410.08	1456.87
No. of 1st Stage Solution	5262.10	1035.26	3607.82	1920.59

*average utilization of vehicles = 0.71

Table 3.8 Summary of the average values of statistics of the sample problem when $n = 15$, $k = 2$ and $|\Xi| = 3$

Difference to the Original Algorithm of Sample Size, $N = 50$	First-stage Cut $T=PDE+FC(0.10)$	Second-stage Cut $T=PDE+SC(0.10)$	Both-stages Cut $T=PDE+BC(0.05,0.10)$
<u>Optimal Value (Z_T)</u>			
- $\bar{X}(N), S(N)$	0.005, 0.011	0.001, 0.006	0.002, 0.006
- 95% Confidence Interval	± 0.003	± 0.001	± 0.001
- % of solutions equal to the solutions obtained by PDE	0.66	0.94	0.82
- Max difference to PDE	0.04	0.05	0.03
<u>Solution CPU Time (sec)</u>			
- $\bar{X}(N), S(N)$	-0.69, 0.26	-0.37, 0.19	-0.54, 0.32
- 95% Confidence Interval	± 0.06	± 0.04	± 0.07
<u>No. of Iterations</u>			
- $\bar{X}(N), S(N)$	-0.72, 0.23	-0.37, 0.16	-0.59, 0.21
<u>No. of B&B Nodes</u>			
- $\bar{X}(N), S(N)$	-0.73, 0.20	-0.34, 0.14	-0.58, 0.18
<u>No. of 1st Stage Solution</u>			
- $\bar{X}(N), S(N)$	-0.81, 0.26	-0.31, 0.16	-0.68, 0.14

Table 3.9 Summary of the difference between the PDE algorithm and enhanced algorithms with different fast fathoming techniques added when $n = 15$, $k = 2$ and $|\Xi| = 3$

larger chance to fathom the optimal solution. However, the deviation from the optimal value obtained by the PDE algorithm is only 0.5% on average and 5% of maximum. As a contrast, second-stage fathoming is applied after obtaining the first stage solution. Therefore, it only generates 0.1% difference to the optimal value of the PDE algorithm. However, it can reduce approximately 30% of solution time. It is because no second stage constraints are added and resolved to the fathomed problem. And, it is the main reason for the reduction of the solution time.

3.5.3 Summary of Computational Results

From the simulation of one scenario of the problem, we demonstrate that each technique can contribute different improvements to the solution time while maintaining the solution close to optimal (within 1% on average). However, solving one scenario is not enough to solve the effectiveness of the techniques since the complexity of solving vehicle routing problem with soft time window is highly dependent on the problem structure. Problems with longer average length of time window or lower waiting/late cost are relatively easy to solve. Also, the utilization of the vehicles (overall demand/overall capacity) also affects the solution time. A high utilization problem is relatively easier to solve than the low one since the number of combinations of the assignments of orders to vehicles are less. Therefore, we can only conclude that these techniques are effective under this scenario. Further analysis should be on different scenario settings to explore the effect of different values of waiting/late cost, lengths of time window, and utilization of vehicle.

Another issue is the parameters setting of different techniques. In our analysis, we did not show the relationship between the values of the parameters to the solution times and optimality. With further analyses, we can choose a suitable parameter setting on different problem structure such that the algorithm can effectively solve the problem.

3.6 Concluding Remarks

To summarize, in this chapter, we describe a vehicle routing problem with time window where the travel time between any delivery points is a discrete stochastic variable. We first model the problem as a 2-stage stochastic integer programming with recourse and propose a branch-and-cut algorithm to solve the problem. Computational results show that this algorithm is not capable enough to solve larger size of problem because of long solution time. Then, we try to introduce some side heuristics to incorporate to the original algorithm. Computational results show that each of these heuristics can help to reduce solution effectively while keeping the solution very close to optimality.

Chapter 4

System Features and Implementation

4.1 Introduction

The development and application of *Information Technology* (IT) has been explosive in recent decades. More businesses start to use information technology tools to smooth their daily operations. With the use of the IT tools, companies can not merely save their operating costs by reducing paper works and human resources, but can also create value to the customers by providing better service level. Many IT tools used in business are decision-support tools. By combining some optimization, data mining or artificial intelligence techniques with the information, decision-support tools can be built to facilitate business decisions making. Because of the trend of globalization of the commercial environment, companies nowadays face customers and suppliers from all around the world. Business decisions now involve a lot of data and information. Evaluating different solutions in order to make the “best” decision is no longer effectively handled by human beings. Therefore, decision-support tools cannot be ignored in any worldwide company in the 21st century. Examples of the application of decision-support tool include 1) using the *Enterprise Resource Planning* (ERP) System for inventory control and demand forecasting. 2) using *Geographical Information System* (GIS) for sales analysis and delivery planning.

In this chapter, we describe a map-based decision-support system, known as *Vehicle Assignment and Navigation System* (VANS), to facilitate the routing and delivery planning decision in Hong Kong. This system consists of a map-based interface to let users visualize the geographical information through a map. Moreover, the system can support several decision-support tools including shortest path finding, time-dependent cheapest path finding, vehicles scheduling and sequencing using real physical distance, etc..

Content of the reminder of this chapter is outlined as follows: In Section 4.2, we briefly describe some specially features in the system including the interface and data management. The decision support tools are described in Section 4.3 and the system implementation is stated in Section 4.4. We will conclude by discussing some further developments of the system.

4.2 System Features

In this section, we focus on some special features of VANS categorized into the following two aspects:

- Map-based interface and network model
- Database management and query

4.2.1 Map-based Interface and Network Model

VANS provides a map-based interface for the users to identify the geographical information to support the decision, data input and analysis. The concept of using digital maps as information representation and analysis started from 30 years ago. At the beginning, research on GIS is to answer the questions how to assemble,

store, manipulate and display the geographical referenced information by a computer-based system. The difference between a GIS map and a paper map is a GIS map combine many *layers* of information that can be selected and displayed under the user preference. The new trend of GIS research is to make use of GIS on some decision supports such as location selection and route finding [11, 45, 54]. We implemented this concept and use a map-based interface in VANS.

(1) Map Structure

The maps provided in VANS are divided into 2 levels as described in Section 2.4. One map of whole Hong Kong with 891 nodes and 1938 road segments is shown. Another map concerns with the street level information. In this map, building information is shown together with the road information. User can select the “combo box” at the top of the map to choose different districts to display. Figure 4.1 shows the 2 maps in the system.

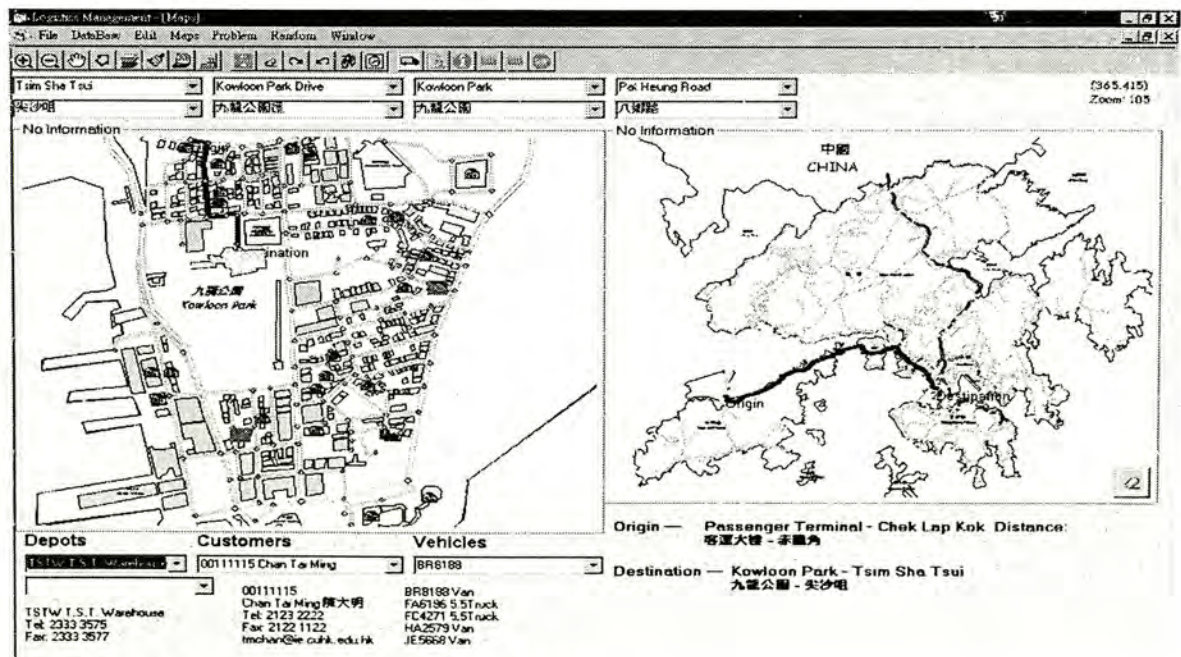


Figure 4.1 Map-based interface of VANS

(2) Point-and-click location indicator

In the map-based interface, users can click on the map to show the referenced

information such as coordinates, area, length and height of a geographical object. At the same time, when the cursor moves around the map, the name of the geographical object will be displayed at the top of the corresponding maps. On the other way round, a series of “combo boxes” is provided at the top of the maps to let user to search for the position of the geographical object using their names in both English or Chinese. The corresponding item will be highlighted on the map.

(3) Network Model

VANS uses a graph containing nodes and arcs to represent the underlying road network. Each point on the map representing the conjunction of the road segments corresponds to a node of the graph. Each link between two points that is the road segment corresponds to an arc of the graph. In the network model, we also consider the direction of the traffic corresponding to the road segment. For instance, a two-way road is represented by two parallel arcs in opposite direction.

(4) Bilingual Geographical Information

Geographical information in VANS, including names of the streets, buildings and districts, can be visualized in both English and Chinese. Having bilingual geographical information built into logistic-support software, the delivery schedule can be reported in either Chinese or English in order to help drivers easily to understand the address of the delivery points. This feature is especially critical in Hong Kong’s bilingual culture. Since a lot of English names of roads or buildings are only the pronunciations of the Cantonese names, this practice leads to a lot of mislocation of the places. Therefore, Chinese address can really help the drivers with minimal English standard to do the delivery quickly and

accurately.

4.2.2 Database Management and Query

There are two main data sources of VANS. One is the map-related data and another is the company-related data. They will be discussed individually as following.

(1) Map-related Data

A relational database is used to represent the map data in VANS. In our network model, each district is represented by an individual graph. Therefore, each district is given a unique ID. A main table is established to store the information of all the districts including the main graph of Hong Kong. Under the main table, each district is represented by a collection of tables. These tables include the information of the vertices (junctions of the road segments), arcs (road segments) and buildings, etc., located in that district. By changing the pointer to different districts, VANS can retrieve the information about the selected district. This design can facilitate the network model used in TCSP2N calculation. Up to this point of time, VANS does not allow users to update or amend the map-related data. Users can only enquire the geographical information from the map-based interface.

(2) Company-related Data

Company-related data, which includes customers, orders, vehicles, depots, products and inventory information, is available in VANS. Users can choose to establish the company database directly in VANS or use an *Electronic Data Interchange* (EDI) link to import data to VANS from an external source.

There is a necessary connection between the company database and map data in order to provide the visualization of company information of the map. In the map database, building is identified a unique ID and is assigned to a node for network calculation. However, in company database, address information is usually represented by a piece of string. This makes the address cannot be identified in the map database. Because of this reason, VANS provides a function that allows users to add the customer information through the map-based interface. Users only need to point and click on the building and the road segment of the digital map, the address of a new customer can be automatically generated. At the same time, the building ID and the corresponding node ID of the customer are stored. This function helps to solve the difficulty in transferring the “word address” to the “system address”. Figure 4.2 shows the interface of this function.

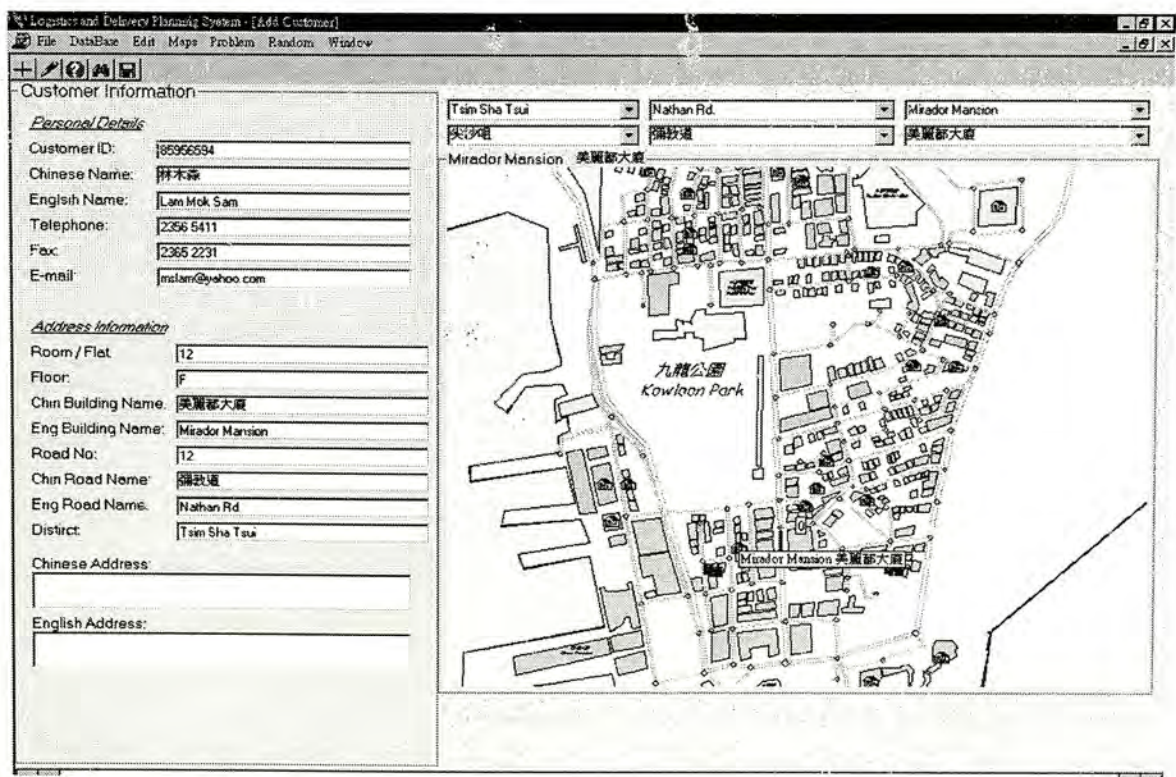


Figure 4.2 Interface for adding new customers

For the products, inventory and orders database, VANS also provides

functions for users to update and add the information. For a delivery system, order handling is an important function. As this result, an interface is available for entering the information of a purchase order. This interface connects with products and customers database. By giving the information of the customer such as customer ID, names or telephone number, the system automatically returns the location, address and related information on the screen. Users can also choose the ordered items stated in the purchase order through product name or ID. By simply typing the quantity of the item ordered, delivery date and time, a commercial invoice can be generated. Also, the weight, volume and amount of the order are also automatically calculated. Since delivery planning requires the consideration of the weight and volume of the goods, this information is essential for route evaluation. Figure 4.3 shows the order-adding interface.

Product ID	Item Description	Quantity	Unit	Unit Price (\$)	Amount (\$)
001226	澳洲純種絲苗5公斤裝	1	Pack	48.3	48.3
003668	綠泉檸檬茶355毫升	20	Can	3.6	72
52237	保利脫藍奶250毫升	3	Pack	4.9	14.7
126301	李錦記VSOP干邑	1	Bottle	263	263
20007	萬應生薑酒450克	2	Pack	6.9	13.8
	萬應生薑酒355毫升				
	萬應生薑酒450克				
	綠泉檸檬茶355毫升				
	澳洲純種絲苗5公斤裝				
	澳洲雙羊百搭米5公斤裝				
	藍妹啤酒640毫升樽裝				
				Total Amount (\$)	463.1

Figure 4.3 Interface for adding new orders

4.3 Decision Support Tools

As a delivery planning support system, VANS offers two major decision-support tools, namely route finding, and delivery planning and scheduling to facilitate

daily delivery operation. The following subsections will discuss these components one by one.

4.3.1 Routing Finding

Route finding is a function to suggest a suitable path from one point to another. The evaluation of the path can be based on distance, time or a combination of both. Up to this point of time, users can find the deterministic shortest path and the time-varying shortest path, as described in Chapter 2, to do the evaluation.

For the deterministic shortest path, the system uses Dijkstra algorithm [24] for calculation. Real distance and travel direction of the road are considered. All parameters are assumed to be fixed in this case. Therefore, the best path obtained in this evaluation should be the same for any departure time of a day. For the time-varying shortest path, the system incorporates the algorithms described in Chapter 2 for computation. The travel time of a road segment is a function of time beginning traversing that segment. Therefore, this calculation can take various travel times at different point of time of a day into consideration. Therefore, the best path between two points at different time of a day can be different in this calculation.

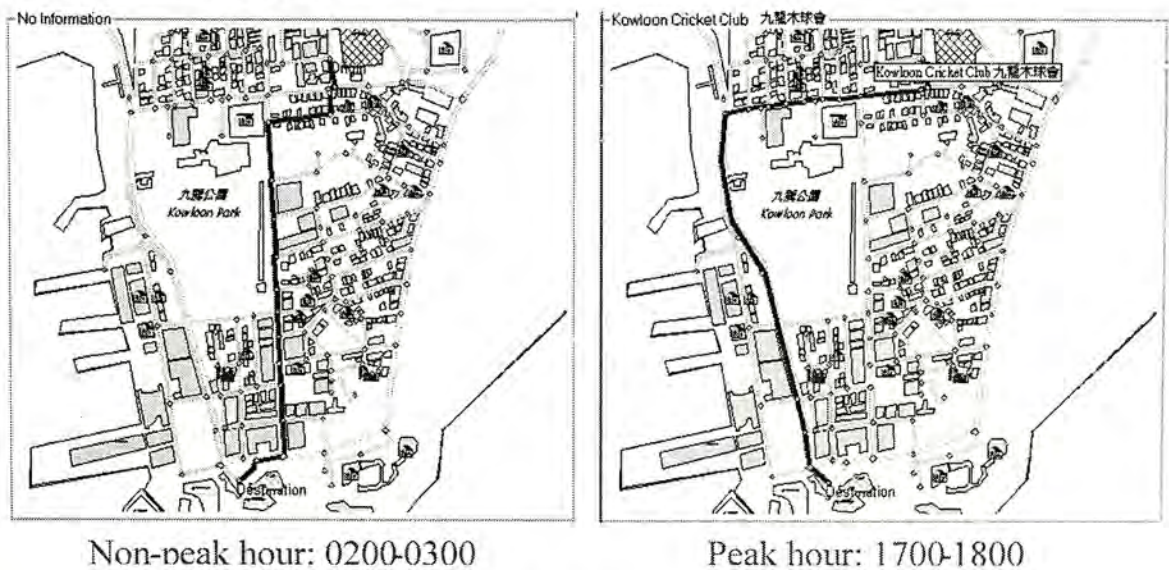


Figure 4.4 Time-varying shortest path at different time intervals

Figure 4.4 demonstrates the difference of the best path between different time intervals of a day. In this case, the starting point and the ending point are Kowloon Cricket Club and the Hong Kong Space Museum in Tsim Sha Tsui respectively. When the departure time starts at non-peak time, say 0200 to 0300, the best path goes straight down the Nathan Road. However, if the same evaluation starts at peak time, say 1700-1800, the best path suggests using Austin Road and Canton Road. The distance travelled by the latter path is obviously longer but is still chosen by the system. This is because the traffic congestion at Nathan Road at peak hours. Therefore, the system suggests using a longer path since the cost function in the evaluation is a linear combination of the distance cost and time cost as described in Equation (2.1).

After the computation of the shortest path, a route report is generated with full details of each road segments past through, including names, distances and travel times, in the travelling sequence of the path.

The main application of the shortest path calculation is to guide the dispatcher to find out the best path. After unloading at one point, dispatcher can use route-finding function to obtain the best path going to the next unloading point. Using the time-varying shortest path, dispatcher can also understand the traffic conditions of the road network and choose the best path while satisfying the time window of the next order.

4.3.2 Delivery Planning and Scheduling

In order to enhance the resource utilization while maintaining a reasonable service level to the customers, VANS provides a delivery-planning function that suggests ways to assign orders to vehicles and schedule the stopping sequence accordingly. After the users inputting all information of orders, vehicles and inventory, VANS will collect all the data and perform an optimization-based scheduling. A Clarke-Wright [15] saving heuristic algorithm is used for this function. The objective of the problem is to find a set of feasible routes with minimum operating costs while satisfying all orders' time windows. One of the features of the algorithm is that time-dependent travel time is used. This can avoid violation of orders' time windows caused by deviation of travel times of a day. The details of the algorithm are described in Appendix A.

After running the algorithm, a set of delivery routes will be obtained. A delivery report is generated and it states the delivery routes of each depot. The delivery sequence, address and estimated arrival/departure time of each route. For each delivery point, the name of the items to be delivered, quantities and amounts are listed. This report helps the dispatchers to collect the goods in the warehouses and manage the time of delivery.

In addition to the textual delivery route report, VANS also provide a nice feature to let the dispatchers to visualize the delivery sequence of the route. After obtaining the delivery schedule, user can choose any one of the routes and show it on the map step-by-step. Information of the current and next loading points, estimated arrival and departure time is displayed under the map. This feature can let the dispatcher understand the correct path between each unloading point and have a better image than reading merely the textual report.

Location and Delivery Planning System - [Route Result]			
File DataBase Edit Maps Problem Random Window			
ROUTE ID: TMDC-01			
WEIGHT CARRIED: 619 kg	VOLUME CARRIED: 118 cm cube	WORKING HOUR: 12:09 -- 12:38	
VEHICLE: FD4985	TYPE: 10Truck		
WEIGHT CAPACITY: 10000 kg	VOLUME CAPACITY: 15000 cm cube	AVALIABLE HOUR: 07:30 -- 15:00	
DISTANCE COST: \$1.5 /km	TIME COST: \$0.5 /min	SET UP COST: \$200	
1 ORDER ID: 007979 TIME WINDOW: 12:14 -- 14:06			
CUSTOMER: 00111115 陳大明 Chan Tai Ming		TEL: 2123 2222	
ADR: 尖沙咀天文臺道保誠保險大廈		ARRIVAL TIME: 12:14	
1 124447 澳洲雙羊百格米5公斤裝	13 Packs	DEPARTURE TIME: 12:24	\$49.6
2 000256 絲芬洗髮水400毫升	42 Bottles		\$644.8
			\$1075.2
2 ORDER ID: 002730 TIME WINDOW: 11:21 -- 13:11			
CUSTOMER: 12588848 章孝炎 CHEUNG Hu Yan		TEL: 2510 2500	
ADR: 尖沙咀奇盛中心		ARRIVAL TIME: 12:25	
1 000256 絲芬洗髮水400毫升	69 Bottles	DEPARTURE TIME: 12:35	\$25.6
2 002585 雀巢咖啡250毫升	23 Cans		\$5.2
3 125779 藍妹啤酒640毫升樽裝	66 Bottles		\$13.8
			\$1766.4
			\$119.6
			\$910.8
DEPOT: TSTW T.S.T. Warehouse TEL: 2333 3575			
ROUTE ID: TSTW-01			
WEIGHT CARRIED: 4240 kg	VOLUME CARRIED: 1165 cm cube	WORKING HOUR: 10:06 -- 17:56	
VEHICLE: FC4271	TYPE: 5.5Truck		
WEIGHT CAPACITY: 5000 kg	VOLUME CAPACITY: 7500 cm cube	AVALIABLE HOUR: 08:00 -- 19:00	
DISTANCE COST: \$1.2 /km	TIME COST: \$0.5 /min	SET UP COST: \$150	
1 ORDER ID: 001105 TIME WINDOW: 10:43 -- 10:52			
CUSTOMER: 19955544 何大一醫生 Dr. Ho Tai One		TEL: 7893 9842	
ADR: 香港中文大學文物館工作處		ARRIVAL TIME: 10:42	
1 001378 莎莉忌廉芝士蛋糕500克	4 Boxes	DEPARTURE TIME: 10:52	\$13.9
			\$55.6
2 ORDER ID: 008889 TIME WINDOW: 11:28 -- 13:02			
CUSTOMER: 00121217 呂四娘 Lui Forth		TEL: 2698 4141	
ADR: 香港中文大學雅禮樓		ARRIVAL TIME: 12:52	
1 003369 道地綠茶355毫升	67 Cans	DEPARTURE TIME: 13:02	\$6
			\$402
3 ORDER ID: 009215 TIME WINDOW: 13:54 -- 14:05			
CUSTOMER: 13001222 任我行 Yam Oi Hang		TEL: 2323 2131	

Figure 4.5 Delivery report

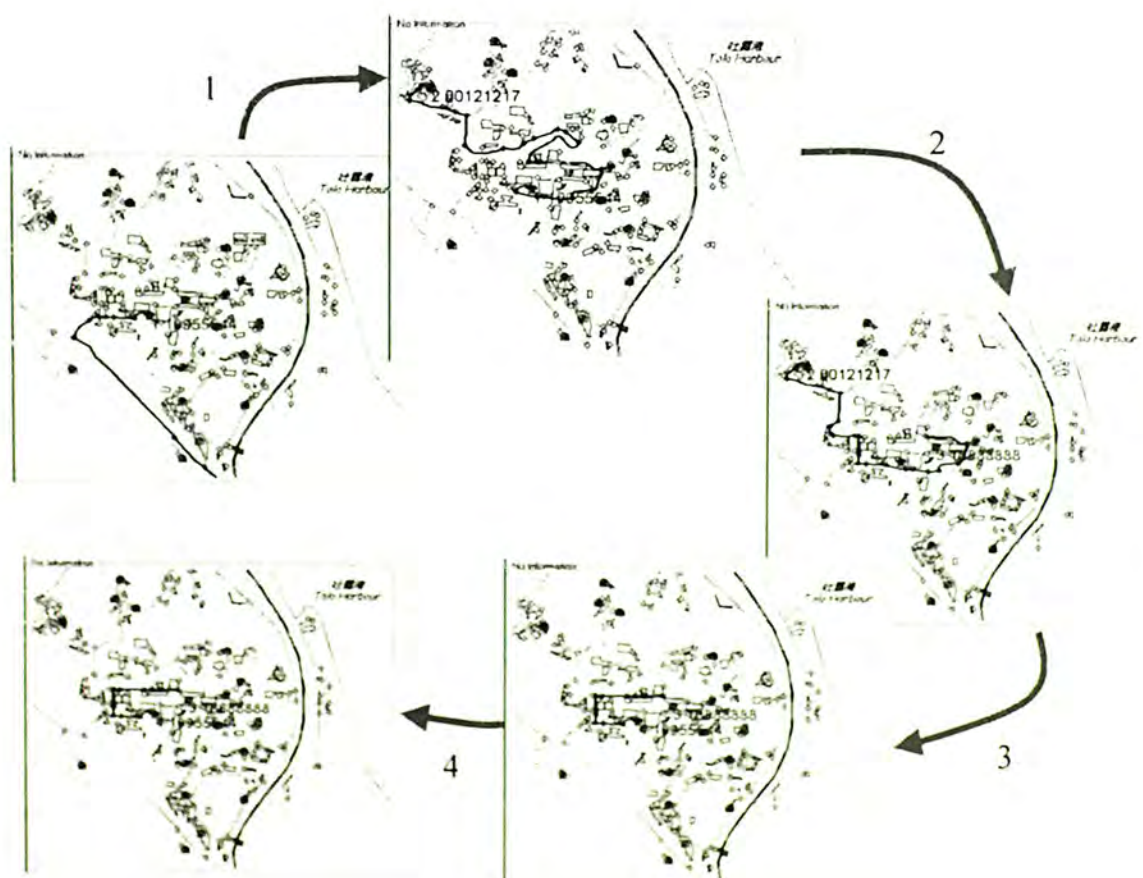


Figure 4.6 Animated route displaying function

4.4 System Implementation

In this section, we briefly describe the implementation of VANS. VANS is a window-platform based system and written in Visual Basic. The GIS tools used are MapX and MapInfo Professional, and the database is managed by Microsoft Access. We will give an introduction of each piece of software used and identify by the relationship between this software in building up the system.

Visual Basic (VB) is a programming language and environment developed by Microsoft Corporation. VB can provide a user-friendly environment to develop the interface of the system. Its event-driven concept allows programmer to manage the events faced by the system, such as mouse click and drag, without difficulties. Also, The object-oriented philosophy of VB also helps us to easily develop the objects or classes, such as customers and vehicles, in the solving procedure. In addition, VB provides a fruitful library codes and applications so that programmer does not need to recreate some common-use applications.

MapX is an active control GIS component developed by MapInfo Corporation. After the installation of MapX, VB can call the routine in MapX to manage the map-related graphics and objects. These routines include various visualization of the map, and selection and enquiry the objects on the map. In VANS, compute must be installed with MapX first to support the management of the map-related objects.

MapInfo Professional is a piece of software developed for mapping and geographical analysis. It is also a product from MapInfo Corporation. The

functions of MapInfo Professional include visualization and analysis of geographical data. By transferring intensive and large amount of information into simple and easy-understanding tables, figure and graphics presentation, users can quickly figure out the problem and make the corresponding decisions. Also, it allows users to create their own maps with specified features and layers. In VANS, MapInfo Professional is used for drawing, layout and design of the network structure of the maps and road networks.

Microsoft Access is a database management system developed by Microsoft Corporation. In VANS, Access is used to manage the map-related data and company-related data. By connecting Access to VB, VANS can provide the data retrieval, updating and enquiring functions.

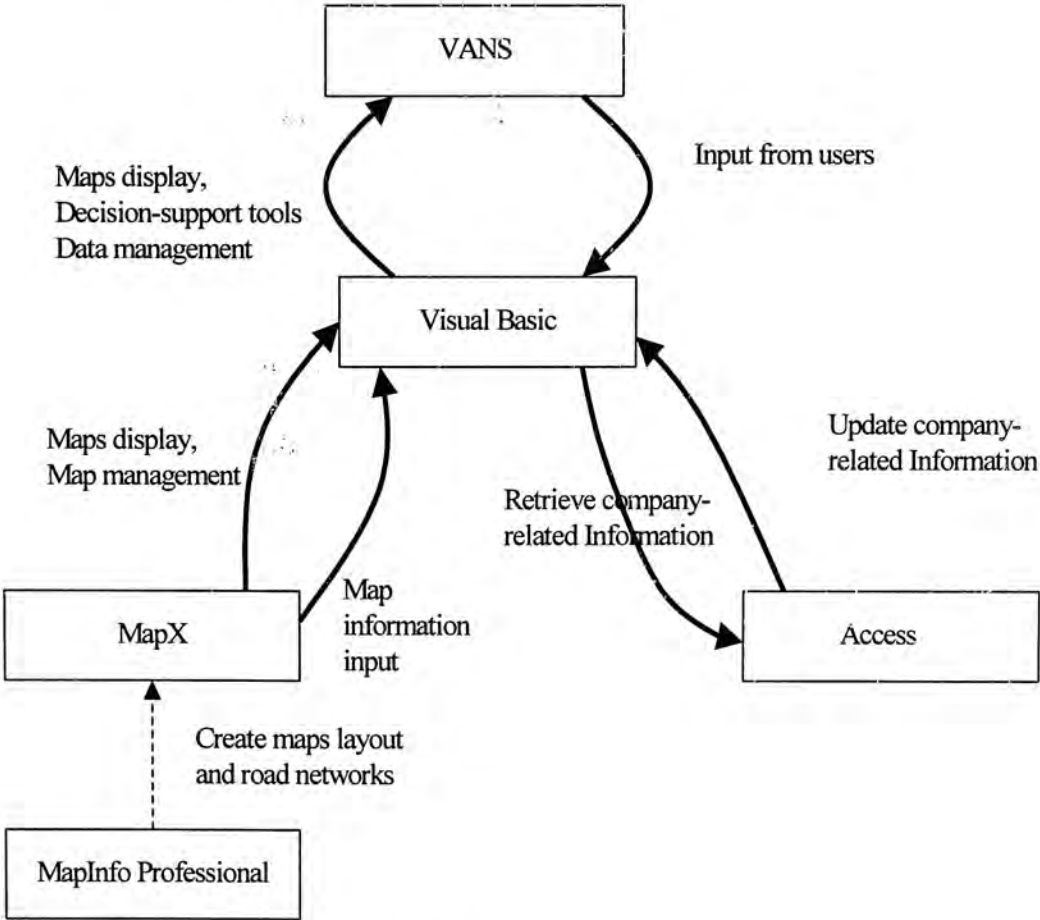


Figure 4.7 Relationships among the software used in VANS

4.5 Further Development

This system is still on going and can have a lot of improvements. On the technical side, *Global Positioning System* (GPS) can be incorporated so that the locations of the vehicles can be reported simultaneously while dealing with real-time orders. Also, at this point of time, real-time and well-quantified traffic information is still lacking in Hong Kong. To improve this, we are going to investigate and analyze the images obtaining from the video camera installed in some of the main links in Hong Kong, and attempt to get some data from the images. For the information transferring between the dispatchers and the control center, some mobile devices, such as cellular phones or *Personal Digital Assistance* (PDA) through the wireless network, can be used to exchange the updated information. This information may include the time-varying shortest path evaluation, re-routing schedule with on-line orders, traffic information or dispatching information. On the service side, an internet-accessed enquiry system will be provided so that customers can have a full picture on the delivery information.

Chapter 5

Vehicle Routing Software Survey

5.1 Introduction

The applications and features of Computerized Vehicle Routing Software (CVRS) have changed rapidly in recent decades. In the past, CVRS was a standalone application installed on an independent computerized machine solving static and simple fleet management problem including assigning stops to vehicles and sequencing them. The interface of the old CVRS was also monotonous as inputting and outputting were all text-based. The objective of applying CVRS that time was to reduce the travelling distance of the delivery process. Nowadays, CVRS is not merely a cost-reducing tool, but is also a strategic planning, customer-service improving and resources utilizing – and profit enhancing system.

In this information technology era, CVRS is widely used in many industries such as manufacturing, food and beverage, service and repair, third-party logistics service providers, etc.. The application of CVRS is wide-spread and commercial companies cannot dispense with its use in order to remain competitive in the industry. Therefore, there are many CVRS providers in the

market. However, the questions of selecting a suitable commercial CVRS for different companies are not identical. It depends on the size, service provided and location of the company. In this chapter, we will first try to figure out and suggest some of the features that are considered essential in CVRS nowadays. Then, we will focus on some of the leading CVRS in the market and elaborate the relatively special characteristics of each product. Finally, we will conclude by addressing the future directions of the CVRS development and the CVRS products sold in Hong Kong. In our study, we focus on several of the most popular CVRS products: 1) ArcLogistics Route by ESRI, 2) ILOG Dispatcher by ILOG, 3) RoutePro Designer by CAPS Logistics, and 4) Paragron by Paragron Software Systems. Company information is provided in Appendix B.

5.2 Essential Features in CVRS Nowadays

CVRS today is versatile and has a lot of product-dependent features. A survey is conducted by OR/MS Today [39] recently on the features of 24 CVRS products in US. In this section, we will look into those features and their benefits to the delivery operations. The first part will focus on some common features that are usually available in current CVRS products. A comparison on the features between traditional products and current products is given. The second part will talk about some special features, which are not found in all products, but are critical to the smooth operation of the delivery process.

5.2.1 Common Features

The followings are the features we determine that must be possessed by a claimed CVRS application.

(1) Roadmap-based interface

At the present time, CVRS products are generally supported by a roadmap-based interface. The details of the roadmaps provided in software are different. For example, in American and European markets, maps provided in leading software are detailed to the street level of each area. From the view of interface of these products, roadmaps only show the information of the road network of the region but without any buildings information. Some companies also provide constant updating of the roadmaps to their clients. Another important GIS capability is geocoding stops from addresses. This means that the system can understand the written address and automatically transform it to geocode of the digital map object. This is a convenient characteristic as mentioned in Chapter 4.

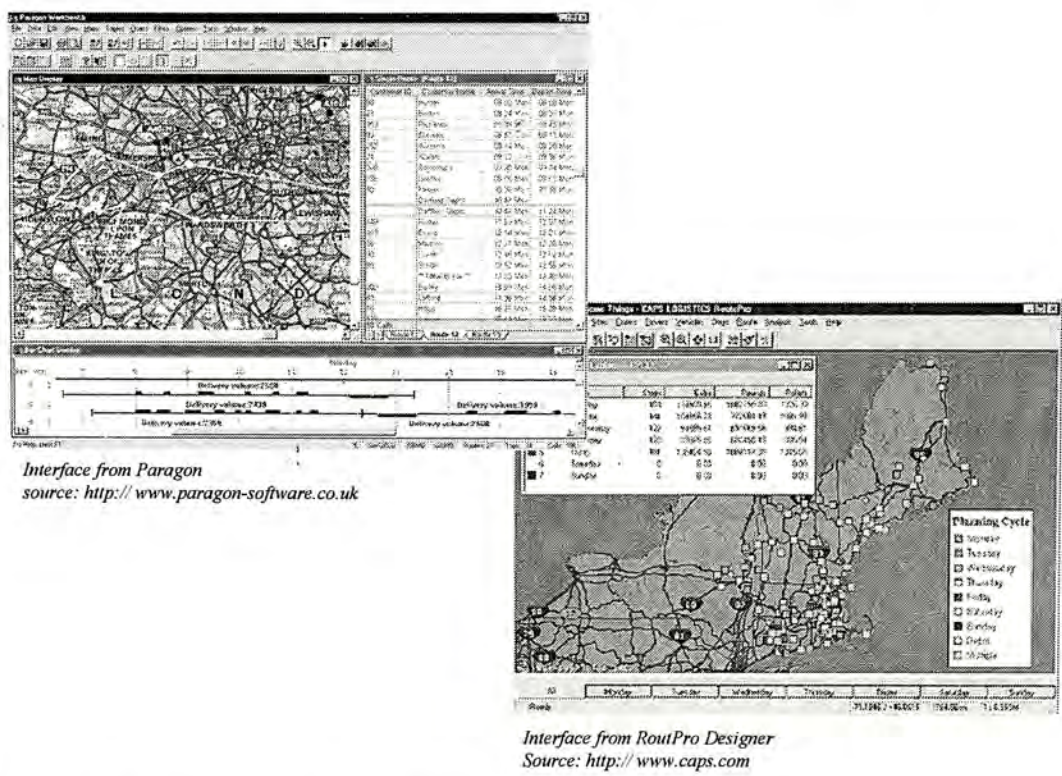


Figure 5.1 Interfaces of some CVRS software

(2) Real distance calculation

Traditional CVRS commonly represents delivery stops by a 2 dimensional Euclidean space and use straight-line distance in calculating delivery schedule. The deviation of the real travel distance to the Euclidean distance can be very large if the road network behind is ignored. Nowadays, with the application of GIS, real travel distance between 2 delivery stops can be quickly and easily computed. Also, road traffic directions are also considered in GIS so that the calculation of the delivery scheduling can reduce errors.

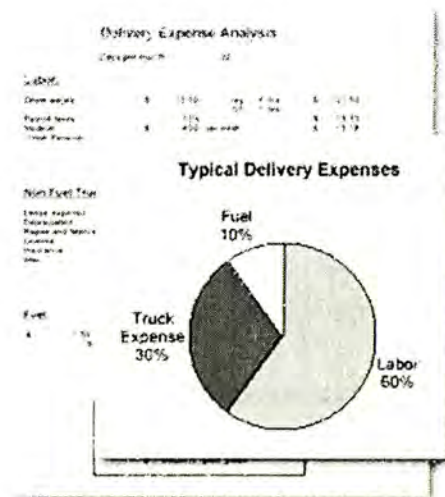
(3) System compatibility

CVRS is no longer an independent software application in this era. As mentioned in the introduction, it can be a strategic planning or resources management system. Therefore, it must be compatible to other software used in the company, e.g. database system ERP system or Microsoft Office. Different companies may use different database to store the information. Since solving delivery schedule must retrieve the information (e.g., on vehicles, orders and customers) in order to facilitate the solution process, compatibility to the data source of a company is an important factor in selecting a suitable CVRS product. After obtaining the delivery schedule, CVRS can export the results to different formats such as Microsoft Word, Excel, Access or postscript. This can allow the users to visualize, amend or further analyze the delivery schedule not only on the CVRS installed machine, but on all general machines.

(4) Results reporting

In the past, CVRS can only give the solution report to the decision maker in textual format. In the report, a list of delivery stops will be given in the order of

delivery sequence. The style and format of the report are rigid and usually cannot be changed easily by the user. At this moment, major CVRS can report the delivery schedule in various ways. Textual report is the basic way to show the delivery details. Users can choose different perspectives to view the solution results. For example, it can be at the point of view of people at operational level (drivers, packers) or managerial level (distribution manager, sales representative). For drivers or packers, the information they needed is the details of the delivery schedule such as desired items or location of each delivery point, arrival time and departure time, and the driving direction. For the distribution managers, they may want to know the overall operating costs and resources utilization so that they can adjust the resources allocation according to the recent performance.



Source: <http://tdinnovations.com/delivery.html>

of each vehicle. On the other hand, some software allows performance analysis. By pulling data from the daily delivery schedules, tables and charts about vehicle, labor and diesel expenses, and capacity utilization are generated. These figures can help the decision maker understand the trend without difficulty, and this convenience cannot be found merely from the textual report.

(5) Interactivity

Solution process in the old CVRS is a one-way process. User starts by giving the problem data, parameters and the system will solve to an “optimal” solution. During the process, user cannot amend anything until the end of the solution process. After the solution is obtained, user can only accept or reject the schedule. Adjustments can only be made manually but the adjusted schedule may not satisfy the problem feasibility. The only way is to re-run the solution process with updated adjustments but this may take a long time. Therefore, flexibility is quite low in the old CVRS.

However, CVRS at this point of time allows users interactively change the delivery schedule. After the CVRS suggests a delivery schedule, user can modify the schedule according to the strategic need by just point-and-drag some objects on interface. For instance, users can interchange the delivery order of two delivery points, assign orders to other vehicles, or add/drop vehicles while keeping some routes unchanged. The system will show the updated costs, driver workload and feasibility of the new schedule. Therefore, users can select to adhere to the old plan or follow the new arrangement. Local re-optimization of part of the problem takes place if needed instead of resolving the whole problem. This feature can let the user change the schedule up to the last minute and enhance the flexibility of resources allocation because of the speedy justification of any new plan. This is also a critical factor in real-time schedule.

(6) Types of schedule

Delivery planning is an intricate problem and the complexity of real situation problem is incredible. Therefore, CVRS actually cannot tackle all the delivery-scheduling problems. In typical CVRS nowadays, it can handle 1) single phase fixed route planning, and 2) multi-period stochastic route planning. Route planning here means normal pickup and delivery planning.

Single-phase fixed route planning is to handle some daily delivery operations or stable distribution over a longer period of time. It is a normal requirement that all CVRS can handle it. Multi-period route planning enables the user to plan schedules for more than one period where delivery frequencies vary between customers. The required amount of customers can also be a stochastic variable. One example is the replenishment of the soft drink selling machines. The system can balance the temporal and spatial factors can come up with an “optimal” schedule.

All the features described above are common found in CVRS products nowadays. Table 5.1 summarizes the main difference between past software and current software.

Features of CVRS	Time	
	up to 1990's	2000's
Interface	text	roadmaps based
Distance used in calculation	Euclidean	real travel distance
System compatibility	standalone	database, ERP connected
Results Reports	Text only	text, maps, graphics, tables
Interactivity	One-way	interactive, real-time
Types of scheduling	single phase, fixed	multi-period, stochastic

Table 5.1 Summary of the feature differences between CVRS of different eras

5.2.2 Advanced Features

After understanding the common features of today's CVRS, we will look into some special features that are thought to be essential in CVRS in this moment or in the future but are not generally found in CVRS products. These features are not only "user-friendly" but also enhance the quality of the delivery solution and also improve the smoothness of the delivery process.

(1) Real time vehicles position tracking

The application of the *Global Positioning System* (GPS) enables the real time tracking of the vehicle position. Through the wireless communication network such as *Global System for Mobile* (GSM), the GPS signal can be transferred to the control panel to display the position of the vehicles. With the use of this feature, users can know sudden events such as accidents and revise the planning accordingly. Another function is that an early estimation can be made from the performance of the routes and highlights any possible missing of time window. Readjustment of the remaining part of the route can be made by the users' insight or suggested by the system.

Beside the operational advantages, the historical data of the vehicle positions can reflect the performance of the drivers to preventing cheating. More than this, the data can be used as a basic estimation of the travel times of some roads so that a time-dependent travel time of the road network can be used in future delivery planning. In our analysis, ArcLogistics Route, RoutePro and Paragron possess this function.

(2) Incorporating real time traffic information

A few of the CVRS applications provide real time traffic information updating to the road database. None of our investigated products provide this function. However, this function is quite important in route planning in some regions with highly varying traffic such as Hong Kong. The first use is that dynamic further planning with a day can use the updated traffic information. Also, re-routing of vehicles on the roads can be decided and sent to the drivers thorough some mobile devices to reduce the chance of vehicles trapped in traffic congestion. This function can be combined together with GPS signal (if available) to achieve a better decision of the amended route. The success of this function depends on whether this real time traffic information is available in the planning region. Cities that have Intelligent Transportation System (ITS) can provide this information comprehensively and speedily. However, a city without this information actually cannot make use of this feature to facility the planning feature.

(3) Travel speeds difference depending on time-of-day

Travel speed of a road segment is not identical throughout the day. Therefore, in an advanced planning situation, it is better to use different travel speeds, known as time-dependent travel time, to represent an actual, or close to actual, traffic condition of a road segment at different points in time. Only RoutePro and Paragron have this feature. The advantage of having this feature is that make a more accurate planning can be done. In this setting, the time window is more accurately satisfied because the travel time between two locations depends on when the traveling takes place. Therefore, the service level can be improved. The

disadvantage of this setting is that the delivery problem with time-dependent travel speeds is more complicated to solve than the version with fixed travel speed. The solution time will be longer and this may make the system unable to react quickly to adjustment. Also, the data of the time-dependent travel speed is difficult to obtain in some regions that make the calculation impossible. Moreover, there is a huge amount of data of travel speeds for all road segments at all time intervals of a day. This leads to the problem of data storage and retrieval in an efficient way.

(4) Strategic planning

Historical delivery data of a company is fruitful information for strategic planning for the future operations. From this data, the customer distribution and orders pattern can be identified. It can help management to decide on the structure of the distribution network as follows:

- Opening and locating a facility such as warehouse and factory
- Allocating inventory within warehouses
- Distributing stock from factories to warehouse
- Increasing or decreasing fleet and crew

In the past, in making the above decisions, management needs to collect a lot of data for analysis. The computation is also difficult to implement because of the extensive amount of data. With the use of CVRS, users can collect the required data easily. Different decision-support tools can be incorporated into the system to perform analysis other than routing. Since delivery cost highly depends on the structure of the distribution network, these decision-support tools are valuable to reduce the overall supply chain costs.

(5) Modeling and algorithmic capability of delivery planning

The delivery planning problems given by industry are much harder to solve than academic problems. The real problems are more complicated and involve much more constraints and objectives. General commercial CVRS products can only handle typical problems with single depot with multiple routes. In the following few lines, we describe some special models that are not commonly handled by commercial CVRS.

If a company operates multiple distribution centers (DCs), it may be beneficial to create routes with product originating from more than one depot. That means one delivery order can be sourced by one vehicle passing thorough different DCs. In this model, it needs to consider the inventory level different products at different distribution centers. The vehicle needs to collect all the required items before arriving to the customers. This also further complicates the problem. Moreover, in multiple DCs situation, the allocation and delivery of inventory between different DCs are another problem to solve. Not many CVRS products can deal with these situations.

Some other constraints are on the driver side. First, the system should be able to match drivers to vehicles according to the licenses they have. Second, lunch hours must be reserved for the drivers. Also, working time of drivers must conform to the regulations of Department Of Transportation and/or Department of Labour. In some countries, drivers must take a rest of a short period time after a specific length of working time. Besides, the workload of drivers should be

balanced to avoid unfairness. Not many systems can handle these additional constraints.

Lastly, order splitting is usually prohibited in commercial CVRS that are generally available. But order splitting in some cases can reduce the number of vehicles used and increase the utilization of all vehicles. Unfortunately, orders splitting may generate difficulty to the receivers such that it may lower customer satisfication. How to balance the cost and the service level is the core of the problem. Also, splitting the orders in a logical way is impossible to be handled by many CVRS systems. Users must specify their own way to do this. Since different companies have their own strategy on this, it is difficult to generalize the model of order splitting. Therefore, not many CVRS products can deal with this.

Table 5.2 summaries the features provided by the investigated products.

Features	ArcLogistics Route ¹	ILOG Dispatcher ¹	Route Pro Designer ¹	Paragon ²
Map-based interface	y	y	y	y
Real distance calculation	y	y	y	y
Real time vehicle tracking	y	-	y	y
Incorporating real time traffic information	-	-	-	-
Different travel speeds of a day	-	-	-	y
Strategic planning tools	ERP version is available	-	Multi- compartments, zone creator, load balancing	Facility selection, distribution between warehouses and factories

Source: ¹OR/MS Today, Feb 2002, ²<http://paragon-software.co.uk>

Table 5.2 Features on investigated CVRS products

5.3 Concluding Remarks

The features described in Section 5.2 are all underdeveloped in current commercial CVRS products. As the computing power is continuously increasing, CVRS can handle more complicated and realistic industrial problem. But still, different companies have their own constraints and delivery model. Because of this reason, it is not able to have a single generalized CVRS product to serve all the situations. Different products may have different strengths. For example, ArcLogistics Route is strong at GIS interface while ILOG Dispatcher is capable in algorithm design and solution quality. Thus, choosing a CVRS product is dependent on the company needs and strategies.

Because of the above reasons, CVRS providers start to divide the whole delivery planning system into several small objects (modules) and let developers access and amend them. Developers can base on the original modules and add constraints or features that they needed. This becomes the new direction of the CVRS products.

Major CVRS providers are located in Europe and America. In Hong Kong, using CVRS in delivery planning is only in an early-developing stage. There are only some small software companies and ArcLogistics Route providing this kind of product custom-tailored to Hong Kong base; that is, the products are installed with Hong Kong road maps and in Chinese. Most of these products provide very elementary functions and nearly none of them provides the advanced features such as real time vehicle tracking and time-dependent travel time. These features

are critical as the traffic condition in Hong Kong seriously varies over time. The difficulties of adding these advanced features to CVRS in Hong Kong are the lack of such information and the incompleteness and inaccuracy of the GPS signal.

In this chapter, we describe some basic features and advanced features of several commercial CVRS products. These features are important to provide users a complete and clear picture of the delivery problem.

Chapter 6

Summary and Further Work

In this thesis, we consider a map-based decision-support system to handle routing and delivery planning problem in Hong Kong. Inside the system, two special decision-support components are designed for the traffic environment in Hong Kong.

The first one is a path finding tool to find out the cheapest path at a transportation road network. The reason of modeling this problem as a Time-varying Constrained Shortest Path Problem (TCSP) is that the time-varying issue of the network is similar to the traffic condition of the road network. Using this tool, we can find a path from one point to another point in the network so that the combination of travel time and distance is minimized. We then extended the original model to our 2-level road network in the system and propose one exact and one heuristic algorithm to solve the problem. The time complexity of the original algorithm is $O(T(n + m))$, where n and m are the number of vertices and arcs of the graph, and $T \in Z^+$ is the maximum time allowed to traverse the path.

The problem solved in the system is to find out a path starting from a point of time, say t_1 , and ending before another point of time t_2 . Since T is an integer in

this model, we need to divide the time interval from t_1 to t_2 into T equal segments. The size of T then becomes a critical factor of the solution time and quality. First, if T is too large, the solution time must be longer since the time complexity is depending on T . In contrast, if T is too small, the travel time function $b(x, y, u)$ may not reflect the correct transit duration at time u since u now is representing a long period of time. The accuracy of the path may be influenced. Therefore, the choice of the size T is needed to further analyze. Because of busy and varying traffic condition in Hong Kong, further work can be on the modeling of expected shortest path at a time-varying and stochastic network.

The second model proposed in this thesis is the Vehicle Routing Problem with Time Window and Stochastic Travel Time (VRPTWST) for delivery planning in Hong Kong. The reason of choosing this model is again that the travel time is difficult to predict because of busy traffic condition. By modeling the travel time between two locations as a discrete stochastic variable, the travel time can be represented by multiple values with different probability. The objective of this problem is to minimize the total operating cost, and total expected waiting cost and late penalty. We proposed an exact branch-and-cut algorithm to solve the problem. Based on this algorithm, a set of heuristics is proposed to improve the efficiency of algorithm. Computational results show that each heuristic can help to reduce the solution time and keep the solution close to optimality.

The solution time of the improved algorithm is still long so that it can only handle a moderate sized problem. Because of this reason, suitable assignments of

orders to vehicles must be made prior to routing problem. The idea is to separate a large problem (many orders and many vehicles) into several smaller pieces of routing problem (moderate number of orders and few vehicles). This approach is very common in solving larger size vehicle routing problem. The application of VRPTWST can be used for fixed route planning in the sense that the delivery pattern will be stable for a relatively long period. Therefore, a longer time can be used to solve the problem. Also, minimizing the expected waiting/late cost is reasonable in a long run case.

The use of VRPTWST is still not sufficient to reflect the complete picture of the traffic condition of whole day. Travel time during the whole day is not identical. Further work can model the travel time between two locations as a time-dependent stochastic variable. That means the probability distribution of the travel time changes over time. This model is further complicated and it is not expected to be solved by branch-and-cut algorithm efficiently. Other approaches such as column generation may be a good approach as we can generate potentially good routes rather than examining all of the possible routes.

For the decision-support system VANS, it now provides a map-based interface, data management and several decision-support tools such as time-varying constrained shortest path and delivery planning. The features provided in VANS are very elementary compared with the commercial products available in the market. The future direction of VANS is in two aspects. First, it will extend the features provided so that it is close to commercial products. Second, it will focus on the special characteristic of delivery in Hong Kong. The

decision support tools provided in VANS must consider the traffic condition, either static or dynamic, of the road network. The solution time should be fast enough to support speedy delivery and on-line adjustment. Moreover, in order to provide time-dependent decision making, we must have the time-dependent travel time data of road network. However, this data is still not available in Hong Kong. One approach can use the *Global Positioning System* (GPS) to identify the travel speed of vehicles on different road segments. By collecting enough data, the average travel speed at different time of different roads can be figured out.

To conclude, this thesis described a decision-support system that is specially designed for the delivery planning in Hong Kong. Besides, two mathematical models are proposed to solve the routing and delivery planning in Hong Kong.

Appendix A

Saving Algorithms used in the System

In the decision support system, a saving heuristic algorithm is used to solve the vehicle routing problem. This algorithm considers multiple depots and time-dependent travel times. The general concept of this algorithm is to assign each order to the closet depot and iteratively insert one feasible order to a route which generate maximum saving. In this model, waiting is allowed but late delivery is not accepted. A general picture of the overall algorithm is outlined as follow:

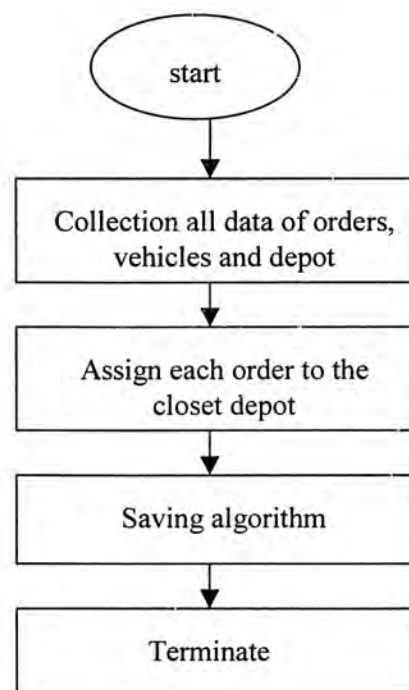


Figure A.1 Overall picture of the saving heuristics algorithm

The core of the algorithm is modified Clark-Wright saving algorithm. Its idea is to insert an order to existing route or create new route by combining two orders. In both cases, the objective is to maximize the cost saved. If no orders can be inserted into any route due to the capacity constraints of vehicle, the algorithm will try to rearrange existing vehicles. It is done by suspending vehicles which have carried a weight up to a threshold, say 80%, of its capacity. Suppose a vehicle carried 85% of weight to its capacity and its route starts at t_1 and ends at t_2 and the working time of this vehicle is t_o to t_d . Then, two dummy vehicles can be created in the follow ways:

1. Starts at t_o and ends at t_1 .
2. Starts at t_2 and ends at t_d .

Then, there will be two new vehicles created. The saving algorithms can be run again until no changed can be found in both orders fulfillment and vehicles arrangement. Figures A.2 and A.3 described the flow of the algorithm.

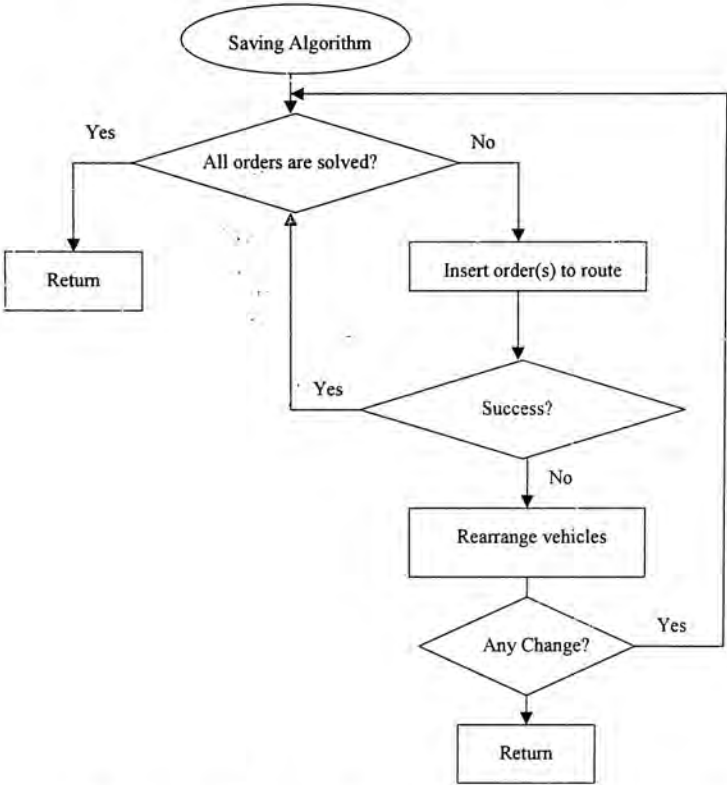


Figure A.2 Procedures of the saving algorithm

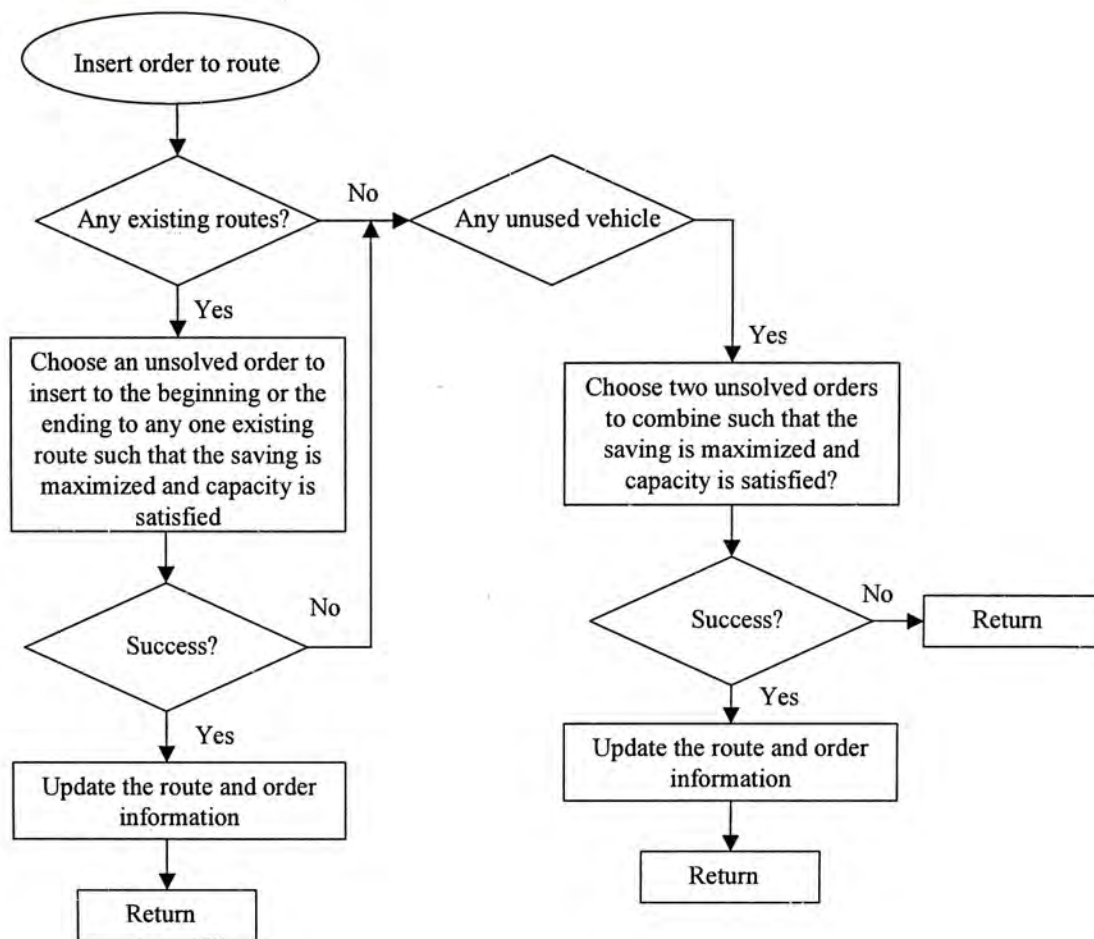


Figure A.3 Procedures of the insert order(s) to route

Appendix B

Background Information of Vehicle Routing Software Company

I. ArcLogisticsTM Route

ArcLogistics Route was a product from Environmental Systems Research Institute, Inc. (ESRI) introduced in 1999. ESRI was founded in 1969 and its current headquarter is located at Redlands, California. The early mission focused on the principles of organizing and analyzing geographic information. During the 1980s ESRI devoted its resources to developing and applying a core set of application tools that could be applied in a computer environment to create a geographic information system. This is what is known today as geographic information system (GIS) technology. In 1981, ESRI launched its first commercial GIS software called ArcInfo. It combined computer display geographic features, such as points, lines, and polygons, with a database management tool for assigning attributes to these features. Originally designed to run on minicomputers, ArcInfo offered the first modern GIS. As the technology shifted to UNIX and later to the Windows operating systems, ESRI evolved software tools that took advantage of these new platforms. This shift enabled users of ESRI software to apply the principles of distributed processing and data

management. Nowadays, ESRI is the worldwide leading GIS software company. Its product includes ArcView, MapObject and ArcIMS, etc.. Currently, ESRI is affording to develop applications using GIS such as Logistics Management.

Source: <http://www.esri.com>

II. ILOG Dispatcher

ILOG Dispatcher is product from ILOG, Inc. ILOG was founded in 1987 to develop and market optimization and visualization software as reusable components of unmatched power and flexibility. ILOG products are used by thousands of developers and tens of thousands of end users in telecommunications, manufacturing, transportation, defense, and other industries. ILOG components have been continually updated to reflect insights gained from over a decade of close collaboration with the world's leading software creators. Its products include ILOG Solver, ILOG CPLEX for optimization, ILOG Scheduler for scheduling task, ILOG Dispatcher for dispatching application. At the same time, it provides solutions to business include e-commerce, finance and transportation.

Source: <http://www.ilog.com>

III. RoutePro Designer

RoutePro Designer is a software application provided by CAPS Logistics, Inc. CAPS Logistics was incorporated in 1979 and started by providing logistics consulting service and developing custom solutions. In 1989, it started to develop software applications to the optimization models of supply chain systems. In 1997, it released three software applications – Supply Chain DesignerTM, TranProTM and RouteProTM solving different supply chain problems.

Source: <http://www.caps.com>

IV. Paragron

Paragron provides a series of vehicle routing and scheduling systems. It is a product launched by Paragron Software Systems plc in United Kingdom. Paragon Software Systems plc specialises in the development and support of vehicle routing and scheduling systems, and has a reputation for providing solutions for strategic and operational use that achieve benefits in a real-world environment across a wide range of sectors. The Paragon system continues to maintain a 60% share of the UK market as a result of its reputation in the industry, and is favoured for its flexibility, ease of use and the comprehensive support provided to its users. Its products include Fleet Controller for vehicle tracking, Fixed Route Manager for designing typical fixed route operation, and Fastnet Distribution Network Planning System for strategic modeling of distribution network, etc.

Source: <http://www.paragron-software.co.uk>

Bibliography

- [1] Ahuja, R.K., K. Mehlhorn, J.B. Orlin and R.E. Trajan (1990), "Faster algorithms for the shortest path problem", *Journal of Association for Computing Machinery*, Vol. 37, pp. 213-223.
- [2] Arabeyre, J., J. Fearnley, F. C. Steiger, and W. Teather (1969), "The Airline Crew Scheduling Problem: A survey", *Transportation Science*, Vol. 3, pp 140-163.
- [3] Ball, M.O., T.L. Magnanti, C.L. Monma and G.L. Nemhauser (eds) (1995), *Network Routing, Handbook in Operations Research and Management Science*, volume 8, North-Holland.
- [4] Bard, J.F., G. Kontoravdi and G. Yu (2000), "A branch-and-cut procedure for the vehicle routing problem with time windows", working paper
- [5] Bastian, C. and A.H.G. Rinnooy Kan (1992), "The stochastic vehicle routing problems revisited", *European Journal of Operational Research*, Vol. 56, pp. 407-412.
- [6] Beasley, J.E., M. Krishnamoorthy, Y.M. Sharaiha and D. Abramson (2000), "Scheduling aircraft landing - the static case", *Transportation Science*, Vol. 34, pp. 180-197.
- [7] Beasley, J.E., J. Sonander, P. Havelock (2000), "Scheduling aircraft landings at London Heathrow", working paper.
- [8] Bellman, R.E. (1958), "On a routing problem", *Q. Applied Math.*, Vol. 16, pp. 87-90
- [9] Bertsimas, D.J. (1992), "A vehicle routing problem with stochastic demand", *Operations Research*, Vol. 40, pp. 574-585.

- [10] Bodin, L., B. Golden, A. Assad and M. Ball (1983), "Routing and scheduling of vehicles and crews: The state of the arc", *Computational Operations Research*, Vol. 10, pp. 62-212
- [11] Bodin, L. and L. Levy (1994), "Visualization in vehicle routing and scheduling problems", *ORSA Journal on Computing*, Vol. 6, pp. 261-268.
- [12] Cai, X., T. Kloks, and C.K. Wong, "Time-varying shortest path problems with constraints", *Networks*, Vol. 29, No. 3, (1988), pp141-149.
- [13] Campbell, A., L. Clarke, A. Kleywegt & M. Savelsbergh (1997), "The inventory problem", working paper, Department of Production and Management Science, University of Vienna, Vienna, Austria.
- [14] Cherkassky, B.V., A.V. Goldberg and T. Radzik (1993), *Shortest path algorithms: Theory and experimental evaluation*, Technical Report pp. 93-1480, Computer Science Department, Stanford University.
- [15] Clarke, G. and W. Wright (1964), "Scheduling of vehicles from a central depot to a number of delivery points", *Operational Research*, Vol. 12, pp. 568-581.
- [16] Cooke, K.L. and E. Halsey (1966), "The shortest route through a network with time-dependent internodal transit times", *Journal of Mathematical Analysis and Applications*, Vol. 14, pp. 493-498.
- [17] Cordeau, J-F, P. Toth and D. Vigo (1998), "A Survey of Optimization Models for Train Routing and Scheduling", *Transportation Science*, Vol. 32, pp. 380-404.
- [18] Corderan, A., A.N. Letchford and J.M. Sanchis (2001), "A cutting plane algorithms for the general routing problems", *Mathematical*

- [19] Corea, G. and V. Kulkarni, "Shortest path problem in stochastic networks with arc lengths having discrete distributions", *Networks*, Vol. 23, pp. 175-183.
- [20] Desaulniers, G. and D. Villworne (2000), "The Shortest Path Problem with Time Windows and Linear Waiting Costs", *Transportation Science*, Vol. 34, No. 3, pp. 312-319
- [21] Desrosiers J., Y. Dumas, M.M. Solomon, F. Soumis (1995), "Time constrained routing and scheduling", in *Network Routing, Handbook in Operations Research and Management Science*, Vol. 8, Ball, M.O., T.L. Magnanti, C.L. Monma and G.L. Nemhauser (eds), North-Holland.
- [22] Desrosiers J., F. Soumis, M. Desrochers and M. Sauve (1986), "Methods for routing with time windows", *European Journal of Operational Research*, Vol. 23, pp. 235-245.
- [23] Desrosiers J., M. Sauve and F. Soumis (1988), "Lagrangian relaxation methods for solving the minimum fleet size multiple traveling salesman problem with time windows", *Management Science*, Vol. 34, pp. 1005-1022.
- [24] Dijkstra, E.W. (1959), "A note of two problems in connexion with graphs", *Numerische Mathematik*, Vol. 1, pp. 269-271.
- [25] Dror, M., (1993), "Modeling vehicle routing with uncertain demand as a stochastic program: Properties of the corresponding solution", *European Journal of Operational Research*, Vol. 64, pp. 432-441.
- [26] Dror, M., G. Laporte and F. Louveaux (1993), "Vehicle routing with stochastic demands and restricted failures", *Z. Operations Research*, Vol.

37, pp.273-283.

- [27] Dror, M., G. Laporte and P. Trudeau (1989), "Vehicle routing with stochastic demands: Properties and solution frameworks", *Transportation Science*, Vol. 23, pp. 166-176.
- [28] Eiger, A., P. Mirchandani and H. Soroush (1985), "Path preferences and optimal paths in probabilistic networks", *Transportation Science*, Vol. 19, pp.75-84.
- [29] Fischetti, M. and P. Toth (1997), "A polyhedral approach to the asymmetric traveling salesman problem", *Management Science*, Vol. 34, pp. 1520-1536.
- [30] Fisher, M.L. (1995), "Vehicle routing", in *Network Routing, Handbook in Operations Research and Management Science*, Vol. 8, North-Holland.
- [31] Fisher, M.L. and R. Jaikumar (1981), "A generalized assignment heuristic for vehicle routing", *Networks*, Vol. 11, pp. :09-124.
- [32] Frank, H. (1969), "Shortest path in probabilistic graphs", *Operations Research*, Vol. 17, pp. 583-599.
- [33] Gallo, G., and S. Pallottino (1988), "Shortest paths algorithms", *Annals Operations Research*, Vol. 13, pp. 3-79.
- [34] Gendreau, M., G. Laporte and J.-Y. Protvin (1999), "Metaheuristics for the vehicle routing problem", Les Cahiers du GERAD, G98-52, Group for Research in Decision Analysis, Montreal, Canada.
- [35] Gillet, B. and L. Miller (1974), "A heuristic algorithm for the vehicle dispatching problem", *Operations Research*, Vol. 22, pp. 340-349.

- [36] Glover, F., D. Klingman and N. Philips (1985), "A new polynomially bounded shortest paths algorithm", *Operations Research*, Vol. 33, pp. 65-73.
- [37] Glover, F., R. Glover and D. Klingman (1984), "Computational study of an improved shortest path algorithm", *Networks*, Vol. 14, pp. 25-37.
- [38] Grotschel, M. and M.W. Padberg (1985), "Polyhedral Theory", in *Traveling Salesman Problem*, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds), pp. 251-306, Wiley, Chichester.
- [39] Hall, R.W., "Vehicle routing software survey", *Operations Research and Management Science Today*, February 2000
- [40] Harker, P.T. (1989), "Use of advanced train control systems in scheduling and operating railroads: Models, algorithms, and applications", *Transportation Research Record*, Vol. 1263: 101-110.
- [41] Hausman, W. and P. Gilmour (1967), "A multiperiod truck delivery problem", *Transportation Research*, Vol. 1, pp. 349-357.
- [42] Hoffman, K.L. and M. Padberg (1993), "Solving airline crew scheduling problems by branch-and-cut", *Management Science*, Vol. 39, pp. 657-82.
- [43] Hung, M.H. and J.J. Divoky (1988), "A computational study of efficient shortest path algorithms", *Computational Operations Research*, Vol. 15, pp. 567-576
- [44] Kao, E.P.C. (1978), "A preference order dynamic program for a stochastic traveling salesman problem" *Operations Research*, Vol. 26, pp. 1033-1045.
- [45] Keenan, P.B. (1996), "Spatial decision support systems for vehicle routing", *European Journal of Operation Research*, Vol. 88, pp. 3-12.

- [46] Kohl, N., J. Desrosiers, O.B.G. Madsen, M.M. Solomon and F. Soumis (1999), "2-path cuts for the vehicle routing problem with time windows", *Transportation Science*, Vol. 33, pp. 101-116
- [47] Kulkarni, V., "Shortest path problem in networks with exponentially distributed arc length", *Networks*, Vol. 16, pp. 255-274.
- [48] Lambert, V., G. Laporte and F. Louveaux (1993), "Designing collection routes through bank branches" *Computational Operations Research*, Vol. 20, pp. 783-791.
- [49] Laporte, G., F., Louveaux and H. Mercure (1992), "The vehicle routing problem with stochastic travel times", *Transportation Science*, Vol. 26, No. 3, pp161-170.
- [50] Laporte, G., F., Louveaux and H. Mercure (1989), "Models and exact solutions for a class stochastic location routing problems", *European Journal of Operational Research*, Vol. 39, pp. 71-78.
- [51] Laporte, G., M. Gendreau, J.-Y. Potvin and F. Semet (2000), "Classical and modern heuristics for the vehicle routing problem", *International Transactions in Operational Research*, Vol. 7, pp. 285-300.
- [52] Larsen, J. (1999), "Vehicle routing with time windows - finding optimal solution efficiently", working paper, Lyngby: Technical University of Denmark.
- [53] Leipala, T. (1978), "On the solutions of traveling salesman problem", *European Journal of Operational Research*, Vol. 2, pp. 291-297.
- [54] Leung, Y., *Intelligent Spatial Decision Support Systems*, Springer, 1997

- [55] Lin, S. (1965), "Computer Solutions of the traveling salesman problem", *Bell System Technology Journal*, Vol. 44, pp. 2245-2269.
- [56] Liu, Q., H.C. Lau, D. Seah and S. Chong, "An efficient near-exact algorithm for large-scale vehicle routing with time windows", Proceedings of the 5th World Congress on ITS Korea, October 1998.
- [57] Loui, R. (1983), "Optimal paths in graph with stochastic and multidimensional weights", *Comm. ACM*, Vol. 26, pp. 670-676.
- [58] Malandrak, C. and M.S. Daskin (1992), "Time dependent vehicle routing problem: Formulations, properties and heuristic algorithms", *Transportation Science*, Vol. 26, pp. 185-200
- [59] Miller-Hooks, E.D. and H.S. Mahmassani (1985), "Least possible time paths in stochastic, time varying networks", *Computational Operations Research*, Vol. 12, pp. 365-381-215
- [60] Miller-Hooks, E.D. and H.S. Mahmassani (2000), "Least expected time paths in stochastic, time varying transportation networks", *Transportation Science*, Vol. 34, No. 2, pp. 198-215
- [61] Mirchandani, P. and H. Soroush (1985), "Optimal paths in probabilistic networks: a case with temporary preferences", *Computational Operations Research*, Vol. 23, pp.365-381.
- [62] Or, I. (1976), Traveling salesman-type combinatorial optimization problems and their relations to the logistics of regional blood banking, PhD dissertation, Northwestern University, Evanston, IL.

- [63] Orda, A. and R. Rom (1990), "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length", *Journal of Association for Computing Machinery*, Vol. 37, pp. 607-625.
- [64] Osman, I.H. (1993), "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem", *Annals of Operations Research*, Vol. 41, pp. 421-451.
- [65] Petersen, E.R., A.J. Talyor and C.D. Martland (1986), "An introduction to computer aided train dispatching", *Journal of Advanced Transportation*, Vol. 20, pp. 52-60.
- [66] Potvin, J.-Y., C. Duhamel and F. Guertin (1996), "A genetic algorithm for vehicle routing with backhauling", *Applied Intelligent*, Vol. 6, pp. 345-355.
- [67] Powell, W., P. Jaillet and A. Odoni (1995), "Stochastic and dynamic networks and routing", in *Network Routing, Handbook in Operations Research and Management Science*, Vol. 8, Ball, M.O., T.L. Magnanti, C.L. Monma and G.L. Nemhauser (eds), North-Holland.
- [68] Russell, R. and G. Gribbin (1991), "A multiphase approach to the period routing problem", *Networks*, Vol. 21, pp. 747-765.
- [69] Sha, D. (1999), *Time Varying Network Optimization Problems*, Ph.D. thesis, Systems Engineering and Engineering Management Department, The Chinese University of Hong Kong.
- [70] Sigal, G., A. Pritsker and J. Solberg (1980), "The stochastic shortest route problem", *Operations Research*, Vol. 28, pp. 1122-1129.

- [71] Smith, M.E. (1992), "Keeping trains on schedule: On-line planning systems for advanced railroad electronic system (ARES)", *Journal of the Transportation Research Forum*, Vol. 31, pp.17-24.
- [72] Sniedovich, M. (1981), "Analysis of a preference order traveling salesman problem", *Operations Research*, Vol. 29, pp. 1234-1237.
- [73] Solomon, M.M. (1987), "Algorithms for the vehicle routing and scheduling problems with time window constraints", *Operations Research*, Vol. 35, No. 2, pp. 254-265.
- [74] Soumis, F., J. Desrosiers and M. Desrochers (1985), "Optimal urban bus routing with scheduling flexibilities", in P. Thoft Christensen (eds), *Lecture Notes in Control and Information Sciences*, Vol. 59, Springer-Verlag, Berlin, Heidelberg, pp. 155-165.
- [75] Stewart, W.R. and B.L. Golden (1983), "Stochastic vehicle routing: A comprehensive approach", *European Journal of Operational Research*, Vol. 14, pp. 371-385.
- [76] Van Breedam, A. (1995), "Improvement heuristics for the vehicle routing problem based on simulated annealing", *European Journal of Operational Research*, Vol. 86, pp. 480-490.
- [77] Xu, J. and J.P. Kelly (1996), "A network flow-based tabu search heuristics for the vehicle routing problem", *Transportation Science*, Vol. 30, pp. 379-393.
- [78] Zhan, F.B. and C.E. Noon (1998), "Shortest path algorithms: An evaluation using real road networks", *Transportation Science*, Vol. 32, No. 1, pp.65-73

CUHK Libraries



003952785