國立交通大學

運輸科技與管理學系

碩 士 論 文

# 有時窗限制的收送貨問題之研究

# The Study of Pickup and Delivery Problem with Time Window Constraints

研 究 生：林信彥

指導教授：王晉元

中華民國九十三年六月

# The Study of Pickup and Delivery Problem with Time Window Constraints

Student：Hsin-Yen Lin　　　　Advisor：Jin-Yuan Wang

Department of Transportation Technology and Management
National Chiao Tung University

## Abstract

It is important for cargo carriers to dispatch their vehicles efficiently and effectively. There are several advantages for efficient and effective dispatches. First, shortening the dispatching time could save the personnel expenses and make better human resources utilization. Secondly, shortening the travel distance of vehicles can reduce the operation and maintenance costs of vehicles. Moreover, the cargo carrier could serve more customers and increase their revenues due to higher utilization rate. Therefore, it is important to develop optimization models and algorithms for such decision making.

We construct an optimization model for pickup and delivery problem with time windows in this research. This model is based on the dispatching rule, and takes current operation status into account. We apply the over-constrained and under-constrained method to deal with the nonlinear time window constraint. The over-constrained method offer an upper bound (a feasible solution) while the under-constrained method provides a lower bound for the solution. In order to approximate the optimal solution, we use the time window partitioning method to narrow the gap between the upper bound and lower bound.

In addition, we use a small example to verify the accuracy and rationality of our model. We also generate our test instances from Solomon's benchmark test samples for VRPTW to evaluate our solving method. The testing results show that our proposed model and solution techniques are suitable for solving this problem.

**Keywords: Pickup and Delivery Problem, Time Window Constraints, Time Window Partitioning.**

有時窗限制的收送貨問題之研究

學生：林信彥　　　　　　　　　　　　　　　　　　指導教授：王晉元

國立交通大學運輸與科技管理學系碩士班

## 摘　　要

對於貨運業者來說，有效率且有效的車輛派遣有許多優點。首先，有效率的車輛派遣可以減少派遣所需的時間，使得人力資源可以有更充分的利用；而縮減車輛的旅行距離可以減少車輛的維修成本；另外，貨運業者可以因為較好的車輛利用率而去服務更多的顧客以增加利潤。因此，發展一個最佳化模式來協助貨運業者進行車輛派遣的決策是相當重要的。

本研究針對有時窗限制的收送貨問題構建了數學規劃的模式，並且針對非線性的時窗限制式做放鬆與緊縮，使得模式成為兩個線性的混合整數規劃(MIP)模型。放鬆時窗限制式使我們得到一個最佳解的下界，而緊縮時窗限制式可得到最佳解的上界(亦為一可行解)，由於兩種模式都為線性規劃，因此可利用一般的數學規劃解題軟體來分別求解以得到上下界。另外，本研究亦使用切割時窗(Time Window Partitioning)的方式來縮減上界與下界之間的差距，當兩者收斂至重合時，代表已經獲得最佳解，然而切割時窗雖然會減小上下界的差距，也會增加問題的規模，因此使用切割時窗法必須在解的品質與求解時間當中做取捨。

本研究利用一個自行設計的小範例來驗證模式的正確性與合理性。此外，由於在文獻上並沒有發現適合本研究的標竿題庫，因此本研究擷取 Solomon 於 1987 年針對有時窗限制的車輛路線問題(VRPTW)所設計的題庫，做部分的修正來產生所需的測試範例，以用來觀察使用不同的切割時窗寬度時，問題規模成長以及上下界收斂的情形。測試結果顯示本研究的模式及所使用的求解方法適用於求解此問題。

**關鍵字：收送貨問題，時窗限制，時窗切割。**

# 誌　　謝

本論文能夠順利完成，首先要感謝王晉元老師的指導。尤其因為此論文以英文撰寫，而我對於英文論文的用字遣辭與撰寫要領都不甚熟悉，因此必須勞煩老師花費許多時間和精神，重複地給予指導與教誨。此外，老師在生活經驗及未來的方向，也常常給予許多寶貴的意見和經驗分享，在此致上最深的謝意。

感謝中華大學交管所 張靖老師以及交大運管系 韓復華老師在論文進行過程中給予相當多寶貴的意見及指導，最後在論文口試時不吝賜教，使得本論文更臻完備。

兩年的研究生生活，感謝和我在同一艘船上奮鬥的所有夥伴。大師兄、家盛、駿逸、建名、穹林等幾位學長，不論是在各種課業或者是研究上的討論、程式開發與報告撰寫，甚至於搭訕與偷吃的手段，都從你們身上獲得相當寶貴的經驗。另外，感謝猴子、嘉龍、knight、丁丁等同學們，在大師兄的帶領下，陪伴我在保齡球館與100分的關卡一次又一次地搏鬥；在打擊練習場不斷地揮空棒(特別感謝猴子)；在研究室裡分享老師的八卦(同時要承受猴子冷笑話的攻擊)。感謝在計畫中與我同甘共苦的學長姐和學弟妹們，感謝你們的指導與配合，我想要說的是：哈!我先下船摟!你們繼續努力吧(￣▽￣)y~。

除此之外，感謝許多研究所的同學們，則言、廢柴豪、寰爺、依潔、小宇、佳琴、小渣、阿貴、博士宏、gay賢、gay百、大乃、大中…，還有兩個不知道算不算是室友的機車小偉和昱豪，曾與你們一起耍廢或誤入歧途，都使我的研究生活更加多采多姿，也祝福你們未來的路都走的順利。

最後要感謝一直在背後支持我的家人，謝謝你們能體諒我有時很久才回家一次，誠心地感謝在這段日子陪著我的所有人，最後引用一句老話：「要謝的人實在太多了，那就謝天吧!」

林信彥

新竹交大
2004/6/29

# Content

## List of Tables

## List of Figures

# Chapter1 Introduction

## 1.1 Motivation

It is important for cargo carriers to dispatch their vehicles efficiently and effectively. The companies usually face a sequence of uncertain service requests for vehicles dispatching. There are several advantages for efficient and effective dispatches. First, shortening the dispatching time could save the personnel expenses and make better human resources utilization. Secondly, shortening the travel distance of vehicles can reduce the operation and maintenance costs. Moreover, the cargo carrier could serve more customers and increase their revenues due to higher utilization rate.

It is not an easy task for large cargo carriers to manage their fleet dispatching jobs. Normally, there are many factors need to be considered simultaneously. However, it is not feasible for these companies to hire sufficient number of skilled people for this purpose.

Therefore, it is important to develop optimization models and algorithms for such decision making. These methods are often described under banners like vehicle routing, work assignment or fleet management, allow the computer to make recommendations regarding work assignment and truck routing and scheduling [1].

## 1.2 Objectives

The objective of this research is to propose an optimization model for pickup and delivery problem with time windows. This model is based on the dispatching rule, and takes current operation status into account. We also propose an algorithm for solving this model.

## 1.3 Scope

The problem we address requires the fulfillment of a set of customer’s pickup and delivery requests on a single depot network. All requests must be executed without violating the vehicle capacity and the customer time window stipulated at each node.

We consider pickup and delivery as separate jobs. That is, there is no need to execute each pair of pickup and delivery load successively. It is possible to insert any job between each pair of pickup and delivery load. Because the size of fleet is fixed, the objective here is to serve as many requests as possible, while minimizing the total travel cost. We do not consider the possibility of future demands.

## 1.4 Study Flowchart

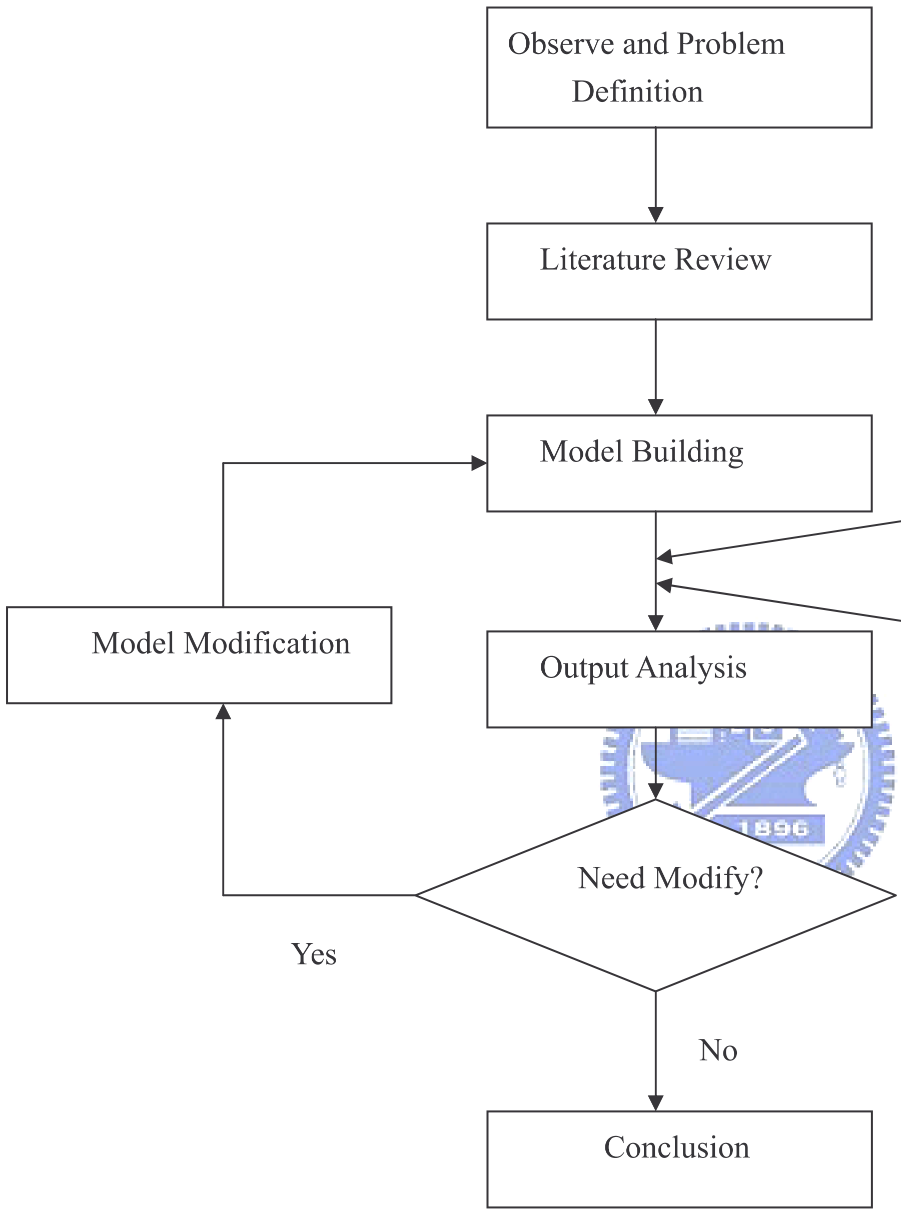


Figure 1.1 Study Flowchart

As shown in figure 1.1, we first observe the operations of a certain company and define the problem. Next we will review relevant literatures of modeling and solving techniques of this problem to know the current status of this research.

Afterward, we define the required parameters and construct the model. Then we propose a method for solving this model. Finally, we generate test instances to evaluate our model and the solution techniques. By analyzing the testing results, we will keep modifying our model until satisfied. Finally, some conclusions are drawn for this research.

# Capter2 Literature Review

In this chapter we review the literatures of general pickup and delivery problem. In section2.1, we review literatures of various pickup and delivery problems. Section2.2 gives an overview of solution approaches that have been proposed.

## 2.1 Review of Characteristics of Pickup and Delivery Problem with Time Windows

The pickup and delivery problems with time windows (PDPTW) involve the satisfaction of a set of transportation requests by a vehicle fleet housed at one or several depots. A transportation request consists of picking up a customer (or load) at the pickup node within a departure interval, and transporting to the delivery node within the arrival interval. The dial-a-ride problem with time windows (DARPTW) is a PDPTW in which the loads to be transported represent people. Therefore, we usually speak of customers or clients instead of transportation requests and all load sizes are equal to zero.

The PDPTW is a special case of vehicle routing problem with time window (VRPTW), which is a high-complexity problem. The solution space is discrete because the decision variables are integer variables. VRPTW was proved to be NP-hard problems [2]. Thus, the possibility of finding a polynomial algorithm is extremely low.

The PDPTW involves a series of constraints: visiting constraints which ensure that each pickup and delivery node is visited exactly once; time window constraints to be satisfied at each node; capacity constraints on each vehicle; coupling constraints stating that a pair of pickup and delivery job must be served by the same vehicle; and precedence constraints imposing that each customer (or load) must be picked-up before the delivery [3].

### 2.1.1 Transportation Requests

A very important characteristic of routing problem is the way in which transportation requests become available. In a static situation, all requests are known at the time the routes have to be constructed. In a dynamic situation, some of the requests are known at the time the routes have to be constructed and the other requests become available in real time during execution of the routes. Most vehicle routing problems are static, whereas most pickup and delivery problems are dynamic. [4]

### 2.1.2 Time constraints

Apart from the vehicle capacity constraints and the intrinsic precedence constraints related to pickup and delivery, side constraints related to time arise in almost every practical pickup and delivery situation. Although time constraints have become an integral part of models for vehicle routing problems, they play an even more prominent role in pickup and delivery problems. Because the most studied pickup and delivery problem is the dial-a-ride problem, which deals with the transportation of people who specify desired pickup or delivery times.

The presence of time window constraints complicates the problem considerably. In the presence of time constraints the problem of finding a feasible pickup and delivery plan is *NP*-hard [4]. Consequently, it may be difficult to construct a feasible plan, especially when time constraints are restrictive. On the other hand, an optimization method may benefit from the presence of time constraints, since the solution space may be much smaller.

### 2.1.3 Objective Functions

A wide variety of objective functions is found in pickup and delivery problem. The most common ones are discussed below.

(A) Minimize duration

The duration time of a route is the total time a vehicle needs to execute the route. Route duration includes travel times, waiting times, loading and unloading times, and breaking times.

(B) Minimize travel time

The travel time of a route refers to the total time spent on actual traveling between different nodes.

(C) Minimize route length

The length of route is the total distance traveled between different nodes.

(D) Minimize client inconvenience

In dial-a-ride systems, client inconvenience is measured in terms of pickup time deviation, i.e., the difference between the actual pickup time and the desired pickup time. Different kinds of functions, linear as well as nonlinear, have been proposed to model client inconvenience.

(E) Minimize the number of vehicles

This function is almost always used in dial-a-ride systems combined with one of the above functions. Dial-a-ride systems are normally subsidized systems for transportation of the elderly and handicapped. Because drivers and vehicles are the most expensive in a dial-a-ride system, minimizing the number of vehicles to serve all requests is usually the main objective.

(F) Maximize profit

This function, which can use all of the above functions, can be used in a system where the dispatcher has the possibility of rejecting a transportation request when it is unfavorable to transport corresponding load. A model based on this objective function should not only incorporate the costs, but also the revenues associated with the transportation of loads. [4]

## 2.2 Review of Solving Techniques for Pickup and Delivery Problem

### 2.2.1 The Static Pickup and Delivery Problem

#### (A) The Static Multi-Vehicle Pickup and Delivery Problem without Time Windows

Cullen, Jarvis and Ratliff [5] propose an interactive approach for the multi-vehicle dial-a-ride problem with homogeneous fleet, i.e., equal vehicle capacities. The problem is decomposed into a clustering part and a chaining part. Both parts are solved in an interactive setting, i.e., man and machine cooperate to obtain high quality solutions. The algorithmic approach in both parts is based on set partitioning and column generation.

The clustering problem, i.e., the problem of constructing and selecting clusters to serve all the clients, can be formulated as a set partitioning problem. Let $J$ be the set of all possible clusters, i.e., seed arcs and assignments of clients to seed arcs. For each $j \in J$, let $c_j$ denote the approximate cost of serving the cluster, and for each $i \in N$, let $a_{ij}$ be a binary constant indicating whether client $i$ is a member of cluster $j$ or not. Furthermore, introduce a binary decision variable $y_j$, to indicate whether a cluster is selected or not. The clustering problem is now to

$$\text{Minimize} \quad \sum_j c_j y_j$$

$$\text{Subject to} \quad \sum_j a_{ij} y_j = 1 \forall i \in N$$

$$y_j \in \{0,1\} \forall j \in J$$

Because the set of all possible clusters is extremely large, a column generation scheme is used to solve the linear programming relaxation of this set partitioning problem.

**(B) The Static Multi-Vehicle Pickup and Delivery Problem with Time Windows**

Dumas, Desrosies and Soumis [6] present a set partitioning formulation for the static pickup and delivery problem with time windows and a column generation scheme to solve it to optimality. The approach is very robust in the sense that it can be adapted easily to handle different objective functions and variants with multiple depots and an inhomogeneous fleet of vehicles.

Desrosies [4] presents a nonlinear- mathematical formulation of the multiple depots multiple vehicle types pickup and delivery problem with time windows, and use Dantzig-Wolfe decomposition algorithm to solve it. The master problem results in the linear relaxation of a set partitioning type model, while feasible routes or columns are generated by a subproblem modeled as a shortest path problem with precedence, time window and capacity constraints.

Haibing Li and Andrew Lin [7] propose a metaheuristic to solve the static pickup and delivery problem with time windows. The approach is a tabu-embedded simulated annealing algorithm which restarts a search procedure from the current best solution after several non-improving search iterations.

William P. Nanry and J. Wesley Barnes [8] present a reactive tabu search approach using three distinct move neighborhoods. A hierarchical search methodology is used to dynamically alternate between neiborhoods in order to negotiate different regins of solution space and adjust search trajectories.

### 2.2.2 The Dynamic Pickup and Delivery Problem

#### (A) The Dynamic Single-Vehicle Pickup and Delivery Problem

Psaraftis [9] extends the dynamic programming algorithm for the static immediate request dial-a-ride problem to the dynamic case. Indefinite deferment of customers, i.e., continuously reassigning service of a customer to the last position in the pickup and delivery sequence, is prevented with a special priority constraint.

The dynamic problem is solved as a sequence of static problems. Each time a new request for service is received, a slightly modified instance of the static problem is solved to update the current route. Obviously, all clients that have already been picked up and delivered can be discarded and the new client has to be incorporated. The starting location the vehicle and the origins of the clients that have been picked up but not yet delivered have to be set to the location of the vehicle at the time of he update.

#### (B) The Dynamic Multi-Vehicle Pickup and Delivery Problem

Psaraftis [10] develops an algorithm for the Dynamic multi-vehicle problem in which vehicles are in fact ships. In this case, the capacity of ports also has to be considered in order to avoid waiting times when loads are to be picked or delivered.

The algorithm is based on rolling horizon principle. Let $t_k$ the current time, i.e., the time at the $k$ th iteration of the procedure. At time $t_k$ the algorithm only considers those known loads $i$ whose earliest pickup time $e_{i+}$ falls between $t_k$ and $t_k + L$, where $L$, the length of rolling horizon, is an user input. The algorithm then makes a tentative assignment of loads to eligible ships.

Powell and Frantzeskakis [11] propose an algorithm which is developed based on network flow representation of the problem. To anticipate future requests, the network is extended with stochastic links. These stochastic links correspond to future uncertain trajectories of vehicles. A maximum profit flow in this extended network not only represents a deterministic allocation of vehicles to load known at $t$=0, but also assigns vehicles to regions in order to serve future requests at minimal cost.

# Chapter3 Model Building

In this chapter we propose a linear model for pickup and delivery problems with time windows (PDPTW). This model considers precedence, coupling, capacity, and time window constraints. In addition, we use the over-constrained and under-constrained method to deal with the non-linear time window constraints. A time window partitioning method is used to approximate the optimal solution.

## 3.1 Definitions and Assumptions

Let $N$ be a set of transportation requests. For each transportation request $i \in N$, a load of size $q_i$ has to be transported from an origin $N_i^+$ (i.e., pickup node) to a destination $N_i^-$ (i.e., delivery node). Notice that $q_i$ is positive for pickups and negative for deliveries. Define $N^+ := \cup_{i \in N} N_i^+$ as the set of all pickup nodes and $N^- := \cup_{i \in N} N_i^-$ as the set of all delivery nodes. Let $V := N^+ \cup N^-$. Define $O$ as the depot. Let $W := V \cup O$. For all $i, j \in V$, let $t_{ij}$ denote the travel time from $i$ to $j$. Furthermore, let $M$ be the set of vehicles. Each vehicle $k \in M$ has a capacity $Q_k$. Note that a node means either a pickup or delivery job. Therefore, we will use node or job to represent a pickup or delivery task.

For each $i \in V$ a time window $[e_i, l_i]$ is introduced denoting the time interval in which service at node $i$ must take place. Given a pickup and delivery plan, we define $T_i$ as the service time of job $i$. Note that the service duration of the pickups and deliveries can be incorporated in the travel times and hence will not be considered explicitly in this research [12], [13]. That is, $t_{ij}$ includes the time vehicles travel from job $i$ to job $j$, and the service duration of job $j$. We define trips as the process of executing a series of pickup and delivery jobs. Step means the process that a vehicle travels form one job to another in its trip. Let $R$ be the set of all steps. The range of steps is from 1 to the maximum number of steps ($H$) that a vehicle can make in a trip. Note that $H$ can be decided by the decision maker in deferent cases. In the extreme case where all jobs can be served by a single vehicle, $H$ should be set to the number of total jobs in order to prevent the missing of the optimal solution.

Each vehicle stops at the last job of its trip. We allow the vehicle to wait if it arrives before the earliest time of the time window. Furthermore, we allow the vehicle to leave if it completes the job before the latest time of time window.

## 3.2 Problem Formulation

The variables of the MIP model are defined as follows:

$Z_{mj}^{k}$ : The binary variable indicates the location of vehicles. $Z_{mj}^{k}=1$ if vehicle $k$ makes a pickup or delivery at node $j$ after $m^{th}$ step of its trip, and 0 otherwise.

$X_{m,i,j}^{k}$ : The binary variable indicates whether vehicle $k$ travels from node $i$ to node $j$ at $m^{th}$ step of its trip. $X_{m,i,j}^{k}=1$ if vehicle $k$ travels from node $i$ to node $j$ at $m^{th}$ step, and 0 otherwise.

$Y_{m}^{k}$ : The continuous variable specifies the load of vehicle $k$ after its $m^{th}$ step.

Using above notations, the formulation of the MIP model is given below:

$$Min \quad \sum_{k\in M}\sum_{m\in R}\sum_{i\in W}\sum_{j\in W} X_{m,i,j}^{k}\cdot t_{ij} + L\cdot(2|N| - \sum_{k\in M}\sum_{m\in R}\sum_{j\in V} Z_{mj}^{k})$$

Subject to:

$$\sum_{m\in R}\sum_{k\in M} Z_{mj}^{k} \le 1 \qquad \forall j \in V \tag{1}$$

$$\sum_{j\in V} Z_{mj}^{k} \le 1 \qquad \forall k \in M, m \in R \tag{2}$$

$$\sum_{i\in V\setminus\{O\}}\sum_{j\in V} X_{1,i,j}^{k} = 0 \qquad \forall k \in M \tag{3}$$

$$\sum_{i\in W} X_{m+1,j,i}^{k} \le Z_{mj}^{k} \qquad \forall k \in M, m \in R, j \in V \tag{4}$$

$$Z_{mj}^{k} = \sum_{i\in W} X_{m,i,j}^{k} \qquad \forall k \in M, m \in R, j \in V \tag{5}$$

$$\sum_{m\in R} Z_{m,N_i^+}^{k} = \sum_{m\in R} Z_{m,N_i^-}^{k} \qquad \forall k \in M, i \in N \tag{6}$$

$$\sum_{k\in M}\sum_{l\in R}(l \cdot Z_{l,N_i^-}^{k}) \ge \sum_{k\in M}\sum_{m\in R}(m \cdot Z_{m,N_i^+}^{k}) \qquad \forall i \in N \tag{7}$$

$$Y_1^{k} = \sum_{j\in V} q_j \cdot Z_{1j}^{k} \qquad \forall k \in M \tag{8}$$

$$Y_m^{k} = Y_{m-1}^{k} + \sum_{j\in V} q_j \cdot Z_{mj}^{k} \qquad \forall k \in M, m \ge 2 \tag{9}$$

$$Y_m^{k} \le Q^{k} \qquad \forall k \in M, m \in R \tag{10}$$

$$(\sum_{m\in R} X_{m,i,j}^{k}) \cdot (T_i + t_{ij} - T_j) \le 0 \qquad \forall i \in V, j \in V, k \in M \tag{11}$$

$$e_i \le T_i \le l_i \qquad \forall i \in V \tag{12}$$

$$Y_m^{k} \ge 0 \qquad \forall k \in M, m \in R \tag{13}$$

$$Z_{mj}^{k}, X_{m,i,j}^{k} = 0 \quad or \quad 1 \tag{14}$$

The first term of the objective function represents the total travel time for all vehicles. L is a large number which penalizes for unserved jobs. That is, the objective is to minimize the total travel time while serving as many as jobs possible.

Constraint (1) ensures that a job can be served once at most. Note that the left hand side of the constraint can be zero, which means the request is not served. Constraint (2) ensures that a vehicle cannot be at more than one node at the end of each step of its trip. Notice that the left hand side of the constraint can be zero in which case either the vehicle is not used or must have finished its trip in a previous step. Constraint (3) requires that each vehicle departs its trip from the depot.

The constraints given by equation (4) and equation (5) establish the link between the variables $Z_{mj}^{k}$ and $X_{m,i,j}^{k}$. If vehicle $k$ at node $j$ after completing $m^{th}$ step of its trip, in which case $Z_{mj}^{k}=1$, equation (4) implies that it can travel to another unserved node from node $j$; otherwise it can not travel to any node originating from node $j$ at $m+1^{th}$ step. In each step, the location of a vehicle is determined by equation (5), namely a vehicle is at node $j$ at the end of $m^{th}$ step if a trip is made to that node at $m^{th}$ step of the process by that vehicle, i.e. $X_{m,i,j}^{k}=1$ for some $i$.

Constraint (6) is the coupling constraint which ensures that each pair of pickup and delivery should be served by the same vehicle. Constraint (7) is the precedence constraint which ensures the pickup must be made before delivery. Note that if the left hand side and the right hand side are both equal to zero, it means vehicle $k$ does not serve request $i$.

The constraints given by equation (8)-(10) are the capacity constraints. Constraint (11) enforces the temporal relationship of consecutive jobs. Constraint (12) specifies the time window constraints. [14]

## 3.3 Modifications of the Model

There are two modifications in this section. First we reduce the number of binary variables. Secondly, we apply the over-constrained and under-constrained methods to deal with the non-linear time constraints. Therefore, we can get a linear model with reasonable number of binary variables. In addition, we use the time window partitioning method to approximate the optimal solution.

### 3.3.1 Variable Redefinition

Two modifications are made in our model. The first modification is the redefining of variables $X^k_{m,i,j}$. In above model, these variables are defined as binary variables. However, the model structure automatically implies that they can only be binary values without any binary constraint. This can be seen by simultaneously considering constraint (2), constraint (4) and constraint (5). Notice that constraint (2) implies that for any $k$ and $m$, $Z^k_{mj} = 0$ for all $j$ except at most one node. Therefore, we can see that if $Z^k_{mj} = 0$, $X^k_{m,i,j} = 0$ for all $i$. Suppose $Z^k_{mj_0} = 1$, then $X^k_{m,i,j} = 0$ for all $j$ except $j = j_0$.

On the other hand, there can be at most one $i$ for which $X^k_{m,i,j_0} \neq 0$, because if $X^k_{m,i_1,j_0} \neq 0$ and $X^k_{m,i_2,j_0} \neq 0$ for some $i_1$ and $i_2$, constraint (4) implies that $Z^k_{m-1,i_1} = 1$ and $Z^k_{m-1,i_2} = 1$ which contradicts constraint (2). Therefore, there is at most one $i$ and one $j$ for which $X^k_{m,i,j} \neq 0$. Constraint (5) then implies that $X^k_{m,i,j}$ is binary since $Z^k_{mj}$ is binary. Therefore, the need for defining $X^k_{m,i,j}$ is binary variable is eliminated. [14]

After redefining $X^k_{m,i,j}$ as continuous variables, the only binary variables remaining in the model are the location variables $Z^k_{mj}$. In this modified model, the total number of binary variables is reduced to $K \cdot H \cdot C$ where K is the number of vehicles, $H$ is the maximum number of steps that a vehicle can make in a trip, and C is the number of jobs which need to be served.

### 3.3.2 Over-constrained and Under-constrained Methods [15]

Because models with nonlinear expressions are much more difficult to solve than linear models, the second modification is to deal with the non-linear time window constraints. Suppose we consider only the end points of the time window of both the first- and the second-job when we set up links between two jobs, then we obtain a solution space that ignores some possible links. We refer to this method as over-constrained method. Suppose we consider only the starting point of the time window of the first job and the end point of the time window of the second job when we set up the link between two jobs, then we obtain a solution space that includes some infeasible links. We refer to this method as the under-constrained method. The over-constrained method makes us to obtain a feasible solution while the under-constrained method provides a lower bound for the solution.

The solid lines in Figure 3.1 and 3.2 show the feasible links excluded and infeasible links included in these two methods. It is not possible to reach job $j$ after leaving at the latest time in the time window for job $i$ (see the dot line in figure 3.1). However, as shown in figure 3.1, it is possible to serve job $j$ after serving job $i$ in the early part of the time window. Comparatively, as shown in figure 3.2, it is possible to reach job $j$ after leaving at the earliest time in the time window for job $i$ (see the dot line in figure 3.2). But it is impossible to serve job $j$ after leaving load $i$ in the later part of the time window for job $i$.

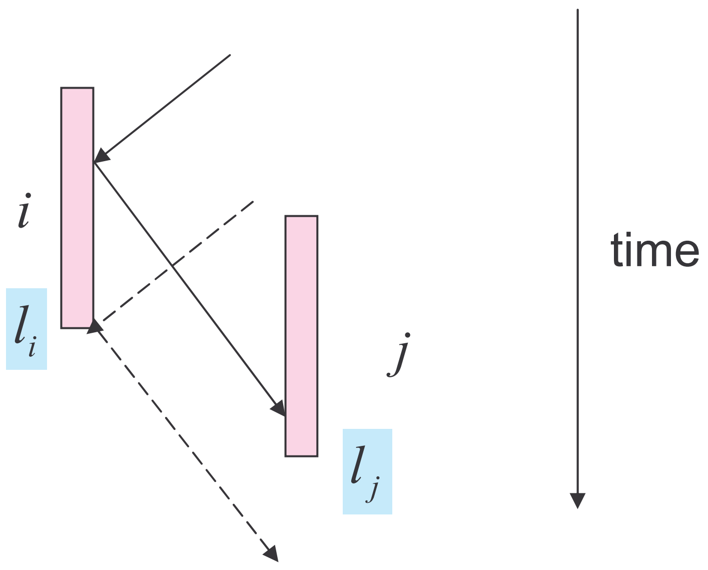


Figure 3.1 Over- constrained Method (feasible links excluded) [15]

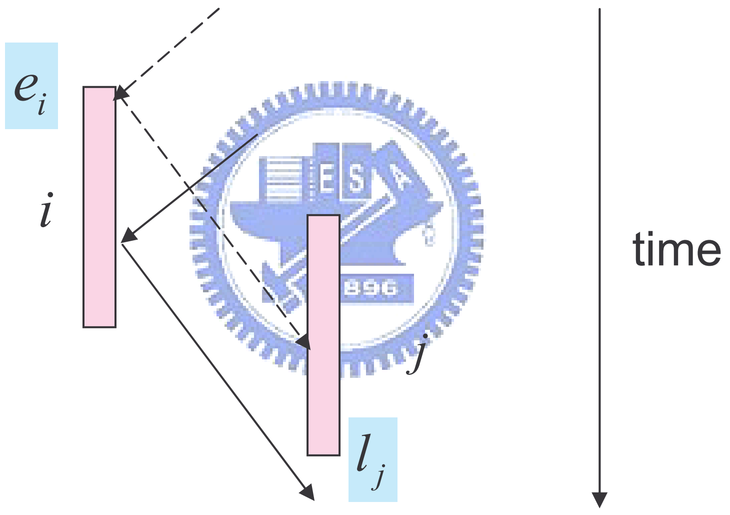


Figure 3.2 Under-constrained Method (infeasible links included) [15]

While using over- constrained method, we replace constraint (11) and (12) with the following:

$$(\sum_{m} X^{k}_{m,i,j}) \cdot (l_i + t_{ij} - l_j) \le 0 \qquad \forall i \in V, j \in V, k \in M$$

While using under- constrained method, we replace constraint (11) and (12) with the following:

$$(\sum_m X^k_{m,i,j}) \cdot (e_i + t_{ij} - l_j) \le 0 \qquad \forall i \in V, j \in V, k \in M$$

If we use the formulation for the under-constrained method, there are some infeasible solutions included in the solution space. While using over-constrained method, some feasible solutions are excluded. As a result, the optimal solution of the formulation by the under-constrained method $Z_{ud}$, over-constrained method $Z_{ov}$, and the global optimal solution $Z_{op}$ can be placed in the following order:

$$Z_{ud} \le Z_{op} \le Z_{ov}$$

After using the over-constrained and under-constrained methods, we can solve the linear models using mathematical programming packages to get $Z_{ud}$ and $Z_{ov}$ [15].

### 3.3.3 Time Window Partitioning [15]

The under-constrained method is used to evaluate the optimality of the feasible solution obtained by the over-constrained method. Thus, the optimal solution is found only when $Z_{ud} = Z_{ov}$. We apply a method for approximating the optimal solution based on this idea.

The time window partitioning method is based on the observation that the gap between $Z_{ud}$ and $Z_{ov}$ is reduced if the time windows are smaller. In general, the larger the time windows, the larger the gap between $Z_{ud}$ and $Z_{op}$, as well as the gap between $Z_{ov}$ and $Z_{op}$. Sometimes the gap between $Z_{ud}$ and $Z_{ov}$ is too large to produce an acceptable solution. Thus, we need to partition the original time windows into several smaller ones. Each partition is considered as a sub-job. At most one of the sub-jobs could be served. The vehicle must enter and leave the same sub-job. By doing this, the number of feasible links excluded by the over- constrained method is reduced, and the number of infeasible links included in the under-constrained method is also reduced.

The issue is to select the width for time window partitioning. Smaller widths may leads to better solutions but also generate larger problem sizes. If the ratio between upper bound and lower bound is unacceptable, it means the selected width is too large. In this case, we select a smaller width and solve the problem again.

We partition the time window in the following way. Suppose the pre-selected width for partitioning is $d$, and the load has time window $[e_i, l_i]$. First, determinate the number of sub-jobs for by taking the smallest integer greater than $(l_i - e_i)/d$. Then, partition the time window into this many parts evenly (see figure 3.3). After doing this, each job is partitioned into several sub-jobs. Each sub-job has its own time window. The set of all sub-jobs can be considered as new set of jobs which need to be served, so the problem size is relatively larger than the original formulation.

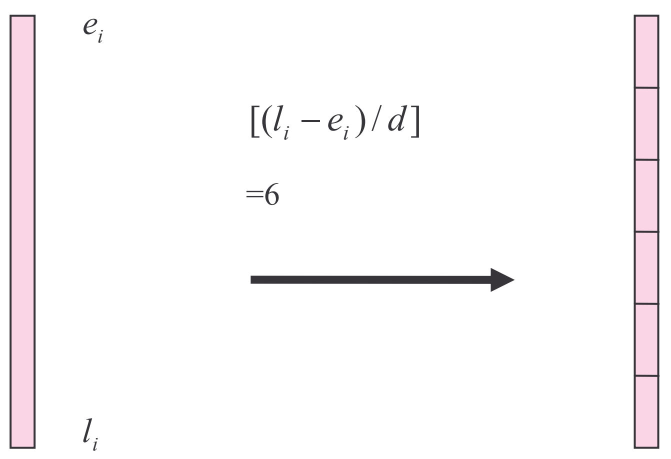


Figure 3.3 Example of Time Window Partitioning

The formulation after time window partitioning can be modified as follows:

$$Min \qquad \sum_{k}\sum_{m}\sum_{i\in S}\sum_{j\in S} X^{k}_{m,i,j}\cdot t_{ij} + L\cdot(2|N| - \sum_{k}\sum_{m}\sum_{j\in S} Z^{k}_{mj})$$

Subject to:

$$\sum_{m\in R}\sum_{k\in M}\sum_{j\in\theta(i)} Z^{k}_{mj} \le 1 \qquad \forall i\in V \tag{1-1}$$

$$\sum_{j\in S} Z^{k}_{mj} \le 1 \qquad \forall k\in M, m\in R \tag{2-1}$$

$$\sum_{i\in S\backslash\{O\}}\sum_{j\in S} X^{k}_{1,i,j} = 0 \qquad \forall k\in M \tag{3-1}$$

$$\sum_{i\in S} X^{k}_{m+1,j,i} \le Z^{k}_{mj} \qquad \forall k\in M, m\in R, j\in S \tag{4-1}$$

$$Z^{k}_{mj} = \sum_{i\in S} X^{k}_{m,i,j} \qquad \forall k\in M, m\in R, j\in S \tag{5-1}$$

$$\sum_{m\in R}\sum_{p\in\theta(N_i^+)} Z^{k}_{m,p} = \sum_{m\in R}\sum_{q\in\theta(N_i^-)} Z^{k}_{m,q} \qquad \forall k\in M, i\in N \tag{6-1}$$

$$\sum_{k\in M}\sum_{l\in R}\sum_{q\in\theta(N_i^-)} (l\cdot Z^{k}_{l,q}) \ge \sum_{k\in M}\sum_{m\in R}\sum_{p\in\theta(N_i^+)} (m\cdot Z^{k}_{m,N_i^+}) \qquad \forall i\in N \tag{7-1}$$

$$Y^{k}_{1} = \sum_{j\in S} q_j\cdot Z^{k}_{1j} \qquad \forall k\in M \tag{8-1}$$

$$Y^{k}_{m} = Y^{k}_{m-1} + \sum_{j\in S} q_j\cdot Z^{k}_{mj} \qquad \forall k\in M, m\ge 2 \tag{9-1}$$

$$Y^{k}_{m} \le Q^{k} \qquad \forall k\in M, m\in R \tag{10-1}$$

$$(\sum_{m\in R} X^{k}_{m,i,j})\cdot(T_i + t_{ij} - T_j) \le 0 \qquad \forall i\in S, j\in S, k\in M \tag{11-1}$$

$$e_i \le T_i \le l_i \qquad \forall i\in S \tag{12-1}$$

$$Y^{k}_{m} \ge 0 \qquad \forall k\in M, m\in R \tag{13-1}$$

$$Z^{k}_{mj}, X^{k}_{m,i,j} = 0 \quad or \quad 1 \tag{14-1}$$

$\theta(i)$ denotes the original job associated with sub-job $i$, $S$ is the set of all sub-jobs. Constraint (1-1) enforces that at most one sub-job can be served for each job, and other constraints remain the same with minor modification regarding part of the summation terms due to the time window partitioning.

We summarize the solving procedure below:

1. Select a series of partition widths (from large to small).
2. Partition the time window using the first unused width.
3. Solve the over-constrained and under-constrained formulations.
4. If the ratio between lower bound and upper bound is acceptable or no smaller width is available, the algorithm stops. Otherwise, return to step2.

# Chapter4 Model Testing and Analysis

In this chapter we test and evaluate our model. We analyze accuracy and rationality using a small example. By checking the solution obtained by our method, we verify the accuracy and rationality of our model. Finally, we generate testing instances to evaluate the performance of our model and solution method.

## 4.1 Accuracy and Rationality Analysis

In the test cases we mention later, we all set $H$ (the maximum number of steps that a vehicle can make in a trip) as the most conservative value. That is the number of jobs which need to be served.

To test the accuracy and rationality, we choose a small instance with four pairs of jobs (four pickup and four delivery jobs) and three vehicles. Table 4.1, 4.2 and 4.3 shows the data we assume in this instance. Notice that job $i$ is a pickup job and job $i$- is the associated delivery job. We assume that the beginning time is 0.

Table 4.1 Vehicle Capacity

| Vehicle number | 1 | 2 | 3 |
|---|---|---|---|
| Capacity | 5 | 7 | 15 |

Table 4.2 Load and Time Window of Each Job

| Job | 1 | 2 | 3 | 4 | 1- | 2- | 3- | 4- |
|---|---|---|---|---|---|---|---|---|
| Load | 7 | 4 | 6 | 8 | -7 | -4 | -6 | -8 |
| Time window | $[3,7]$ | $[2,5]$ | $[1,4]$ | $[5,9]$ | $[7,10]$ | $[8,10]$ | $[6,8]$ | $[9,12]$ |

Table 4.3 Distance Matrix

| | O | 1 | 2 | 3 | 4 | 1- | 2- | 3- | 4- |
|---|---|---|---|---|---|---|---|---|---|
| O | | 1 | 0.5 | 1 | 2 | 1 | 1 | 1 | 1 |
| 1 | 1 | | 1 | 1 | 1 | 0.5 | 2 | 2 | 1 |
| 2 | 0.5 | 1 | | 2 | 1 | 2 | 0.5 | 2 | 1 |
| 3 | 1 | 1 | 2 | | 1 | 2 | 0.5 | 1 | 2 |
| 4 | 2 | 1 | 1 | 1 | | 1 | 0.5 | 2 | 2 |
| 1- | 1 | 0.5 | 2 | 2 | 1 | | 1 | 2 | 1 |
| 2- | 1 | 2 | 0.5 | 0.5 | 0.5 | 1 | | 2 | 1.5 |
| 3- | 1 | 2 | 2 | 1 | 2 | 2 | 2 | | 1 |
| 4- | 1 | 1 | 1 | 2 | 2 | 1 | 1.5 | 1 | |

There are several criteria for verifying the accuracy and rationality:

A. The comparison with optimal solution.

To verify the accuracy of our methods in this instance, we compare $Z_{ud}$ and $Z_{ov}$ with the optimal solution. We check if the optimal solution is between $Z_{ud}$ and $Z_{ov}$, Then, we compute the error ratio of our solution in this instance.

B. Precedence constraints of pair jobs.

We verify if each pair job is served by the same vehicle. We also check if the pickup job is served before the delivery job.

C. Vehicle capacity constraints.

We check if the capacity restriction is not violated at every step of trips.

D. Time window constraints.

Each job has a time window $[e_i, l_i]$. We verify if every job could be executed within its time interval.

The results we obtained from the over-constrained and under-constrained methods are shown in Table 4.4 and Table 4.5.

Table 4.4 Solution Obtained by the Over-constrained Method: $Z_{ov}$ =7

| Vehicle | Trip |
|---|---|
| 1 | $O \rightarrow 2 \rightarrow 2-$ |
| 2 | $O \rightarrow 3 \rightarrow 3-$ |
| 3 | $O \rightarrow 1 \rightarrow 4 \rightarrow 1- \rightarrow 4-$ |

Table 4.5 Solution Obtained by the Under-constrained Method: $Z_{ud}$ =6

| Vehicle | Trip |
|---|---|
| 1 | None |
| 2 | $O \rightarrow 3 \rightarrow 3-$ |
| 3 | $O \rightarrow 2 \rightarrow 2- \rightarrow 4 \rightarrow 1 \rightarrow 1- \rightarrow 4-$ |

According to the criteria, the testing results satisfy our dispatching rules. First, the pair jobs are served in correct sequence and there is no pickup or delivery job in reverse order. Secondly, each vehicle at any step of its trip does not violate the capacity constraints, (see table 4.6, 4.7). Thirdly, with the solution of over-constrained method, every job can be served during its time window (see table 4.8).

Table 4.6 Load at Each Step (Over-constrained Method)

| Vehicle | Capacity | Load at the End of Each Step |
|---|---|---|
| 1 | 5 | $0 \to 4 \to 0$ |
| 2 | 7 | $0 \to 6 \to 0$ |
| 3 | 15 | $0 \to 7 \to 15 \to 8 \to 0$ |

Table 4.7 Load at Each Step (Under-constrained Method)

| Vehicle | Capacity | Load at the End of Each Step |
|---|---|---|
| 1 | 5 | None |
| 2 | 7 | $0 \to 6 \to 0$ |
| 3 | 15 | $0 \to 4 \to 0 \to 8 \to 15 \to 8 \to 0$ |

Table 4.8 Service Time of Each Job (Over-constrained Method)

| Vehicle | Time window of each job | Service time of each job |
|---|---|---|
| 1 | $none \to [2,5] \to [8,10]$ | $0 \to 2 \to 8$ |
| 2 | $none \to [1,4] \to [6,8]$ | $0 \to 1 \to 6$ |
| 3 | $none \to [3,7] \to [5,9] \to [7,10] \to [9,12]$ | $0 \to 3 \to 5 \to 7 \to 9$ |

Notice that the solution obtained by the under-constrained method is not a feasible solution. Time window constraint is violated when vehicle3 travels from job4 to job1. Note that vehicle3 leaves job4 at the time 8.5. It is impossible to serve job1 in time, (see figure 4.1, 4.2). Because of the time window constraint of the under constraint method, the link from job4 to job1 becomes a feasible link (see the dot line in figure4.2).

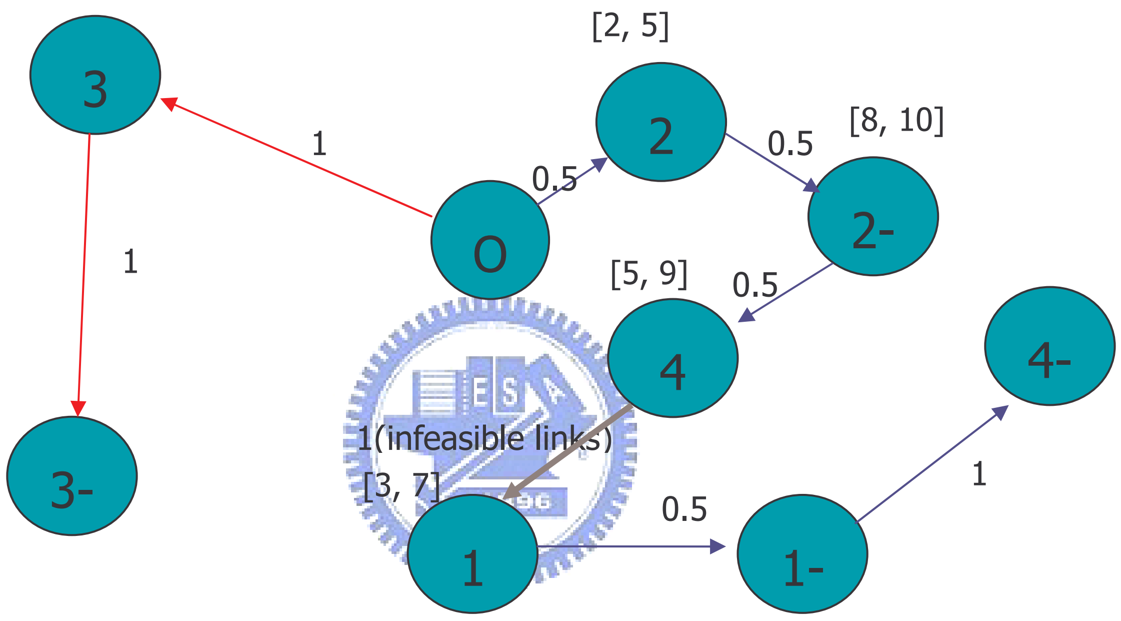


Figure 4.1 Solution of the Under-constrained Method

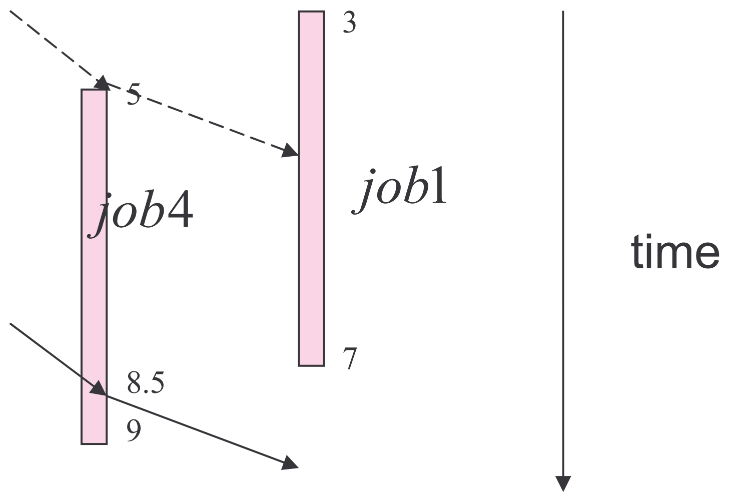


Figure 4.2 The Infeasible Links Included by the under-constrained method

In addition, the optimal value of over-constrained method $Z_{ov}$ is 7, which is kind of close to $Z_{ud} = 6$. There is no need to partition the time window any further in this small example. The solution we obtained is 7, and our error ratio in this example is at most $7 \div 6 \cong 1.167$. In fact, we can obtain the optimal value $Z_{op} = 7$ of this problem by simply observation. Thus, the relationship $Z_{ud} \le Z_{op} \le Z_{ov}$ can be verified.

## 4.2 Test Instance Generation

There is no comprehensive benchmark test sample available for our problem. However, from the literature of VRPTW, there are well-established benchmark test samples for VRPTW by Solomon [16]. The geographical data are randomly generated in problem sets R1 and R2, clustered in problem sets C1 and C2, and a mix of random and clustered structures in problem sets by RC1 and RC2. Problem sets R1, C1 and RC1 have a short scheduling horizon and allow only a few customers per route (approximately 5 to 10). In contrast, the sets R2, C2 and RC2 have a long scheduling horizon permitting many customers (more than 30) to be served by the same vehicle. In this section we present how we adapt Solomon's instances to generate our test instances.

Unlike VRPTW in which jobs have no coupling and precedence constraints, a PDPTW does. If we generate our test instances from Solomon's benchmark instances by randomly pairing up the nodes, it may be impossible to obtain a feasible solution for serving all jobs. For example, if we randomly pair up a pickup job $i$ with time window [200, 300] and a delivery job $j$ with time window [0, 150]. It is impossible to serve job $i$ before job $j$. If this happens, the pickup and delivery pair (P-D pair) we generate would become unmeaningful. Therefore, the issue is how to generate reasonable pickup and delivery pairs.

To avoid the condition we mentioned above, we need to confirm that every P-D pair can be served without violating time window constraint. To do this, we check the time window constraint of over-constraint method for each P-D pair. Our procedure is as shown below:

```
Procedure GENERATE:
$k = 1$
Do until $k$ = number of P-D pairs need to generate
    1. Randomly select two jobs $(i, j)$ to be paired.
    2. If $l_i + t_{ij} > l_j$ return to step1, else go to next step.
    3. Randomly select either $i$ or $j$'s load as pickup and delivery load for both $i$
       and $j$. $k = k + 1$.
```

Because of the time window constraint of over-constraint method is tighter than under-constraint method, the P-D pairs we generate by this procedure will be valid for both two methods.

## 4.3 Performance Evaluation

In this section we first test the quality of the solution without time window partitioning. Then we evaluate the performance of time window partitioning method using different partitioning widths. By observing the tightness of the upper bound and lower bound, we evaluate the effect of time window partitioning method.

### 4.3.1 Performance without Using Time Window Partitioning Method

Using the test instance generation procedure discussed in section 4.2, we generate 20 instances from Solomon's benchmark instances. Instance 1 to 10 are generated from R1 type, and instance 11 to 20 are generated from R2 type .Each instance consists of 2 vehicles with capacity 50 and 10 pickup and delivery jobs. We solve the problem using LINGO8.0 with no special modifications. All tests are run on a desktop computer with 2.4 GHz Pentium IV CPU and 512 MB RAM.

Table 4.9 shows the results of 20 instances without using time window partitioning technique. Because R2 type problems have a long scheduling horizon, the time windows generated from R2 usually have larger width. Therefore, the gap between $Z_{ov}$ and $Z_{ud}$ is larger for instance 11 to 20.

Table 4.9 Result without Using Time Window Partitioning Technique

| Instance Id (From R1) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $Z_{ov}$ | 220 | 205 | 202 | 205 | 270 | 197 | 196 | 197 | 290 | 188 |
| $Z_{ud}$ | 189 | 191 | 179 | 163 | 214 | 177 | 163 | 158 | 254 | 152 |
| Ratio | 0.859 | 0.932 | 0.886 | 0.796 | 0.793 | 0.898 | 0.832 | 0.802 | 0.875 | 0.809 |
| Instance Id (From R2) | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| $Z_{ov}$ | 387 | 248 | 290 | 274 | 258 | 287 | 286 | 228 | 293 | 252 |
| $Z_{ud}$ | 300 | 210 | 217 | 209 | 182 | 246 | 211 | 213 | 259 | 178 |
| Ratio | 0.775 | 0.847 | 0.748 | 0.762 | 0.706 | 0.857 | 0.738 | 0.934 | 0.884 | 0.706 |

Because the solution space of under-constraint method is relatively larger, the time required to obtain the lower bound may be lengthy. In order to reduce the solution time, the solution obtained from the over-constrained method is used as a cut off point for the branch and bound algorithm. A good hurdle value can greatly narrow the searching space for the optimum and reduce the solution time of the under-constrained formulation. The solution times for these instances are all within 8 minutes.

### 4.3.2 Performance of Time Window Partitioning Method

We test 10 instances using time window partitioning method in this section. Each instance is generated from R1 type with 3 vehicles (capacity =50) and 10 pickup and delivery jobs. With different time window partitioning widths (None, 30, 20, 10), we try to approximate the optimal solution of each instance. The ratio of $Z_{ov}$ and $Z_{ud}$ are provided in table 4.10. The number of sub-jobs means the total number of sub-jobs after partitioning the time windows using a specific partitioning width. Note that we stop the solving procedure when any one of the following conditions is reached:

1. The optimal solution is found (ratio =1).

2. The pre-selected partitioning widths are run out.

Table 4.10 Test of Time Window Partitioning Method

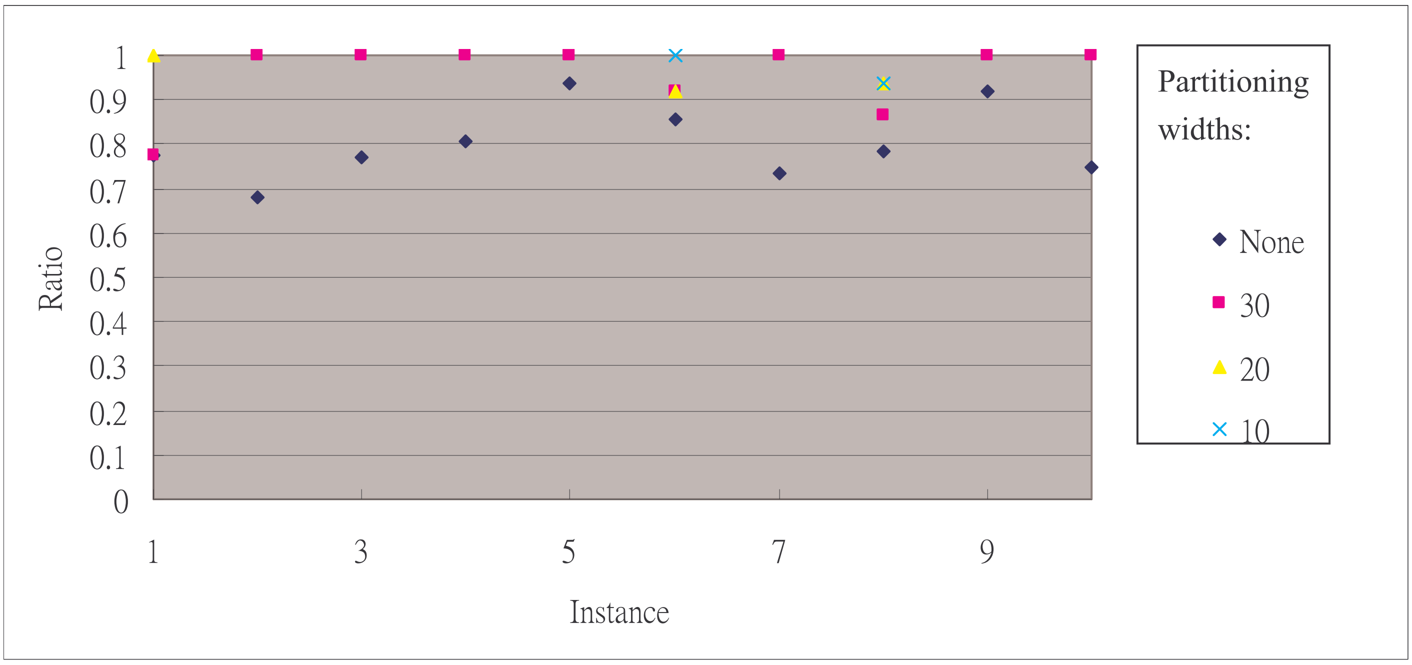
| Instance Id | Partitioning width | $Z_{ov}$ | $Z_{ud}$ | Ratio | Number of Sub-jobs | CPU Time for Obtaining $Z_{ov}, Z_{ud}$ (Seconds) |
|---|---|---|---|---|---|---|
| 1 | None | 214 | 166 | 0.776 | None | 1 , 30 |
| | 30 | 214 | 166 | 0.776 | 11 | 1, 53 |
| | 20 | 214 | 214 | 1 | 18 | 5, 124 |
| | | | | | | |
| 2 | None | 248 | 169 | 0.681 | None | 1, 402 |
| | 30 | 212 | 212 | 1 | 25 | 108, 482 |
| | | | | | | |
| 3 | None | 323 | 249 | 0.771 | None | 1, 853 |
| | 30 | 297 | 297 | 1 | 25 | 307, 967 |
| | | | | | | |
| 4 | None | 266 | 215 | 0.808 | None | 1, 384 |
| | 30 | 266 | 266 | 1 | 23 | 401, 784 |
| | | | | | | |
| 5 | None | 114 | 107 | 0.939 | None | 1, 240 |
| | 30 | 107 | 107 | 1 | 27 | 678, 1055 |
| | | | | | | |
| 6 | None | 160 | 137 | 0.856 | None | 1, 76 |
| | 30 | 160 | 147 | 0.919 | 11 | 5, 24 |
| | 20 | 160 | 147 | 0.919 | 13 | 6, 50 |
| | 10 | 147 | 147 | 1 | 29 | 846, 886 |
| | | | | | | |
| 7 | None | 172 | 128 | 0.736 | None | 1, 294 |
| | 30 | 161 | 161 | 1 | 27 | 216, 1732 |
| | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | None | 255 | 200 | 0.784 | None | 2, 491 |
| | 30 | 255 | 221 | 0.867 | 12 | 3, 31 |
| | 20 | 236 | 221 | 0.936 | 13 | 7, 49 |
| | 10 | 236 | 221 | 0.936 | 26 | 83, 482 |
| | | | | | | |
| 9 | None | 227 | 209 | 0.921 | None | 1, 846 |
| | 30 | 227 | 227 | 1 | 11 | 6, 23 |
| | | | | | | |
| 10 | None | 290 | 217 | 0.748 | None | 1, 1521 |
| | 30 | 290 | 290 | 1 | 14 | 6, 175 |

Figure 4.3 shows that the same series of partitioning widths makes different convergence effect on different instances. Optimal solutions are found on 7 out of 10 instances in early partitioning procedures (with partitioning width 30).

Because the original time window width of each job is not the same, a partitioning width often leads to different number of sub-jobs on each instance. Thus, an instance consists of many jobs with large time window width can be partitioned into more number of sub-jobs even though the partitioning width is the same. It will make the problem size expand quickly. However, it does not necessarily mean that we can obtain a rapid improvement of the solution. Another observation is that these solution procedures consume a lot of computation time when a “too small” partitioning width is used. That is, we expand unnecessary problem sizes for solving to its optimality. Therefore, it is important to construct a good partitioning strategy for different instances to avoid the issues mentioned above.

Figure 4.3 Effect of Time Window Partitioning Method



# Chapter5 Conclusions and Suggestions

## 5.1 Conclusions

In this research, we have constructed a MIP model for pickup and delivery problems with time windows. We use the over-constrained and under-constrained method to deal with the nonlinear time window constraint. The under-constrained method yields a lower bound which is used to evaluate the optimality of the feasible solution yielded by the over-constrained method. Therefore, the models we present to obtain the upper bound and lower bound are both linear mathematical formulations so that they could be solved by using general LP solvers.

The optimal solution is found only when the upper bound ($Z_{ov}$) equals to lower bound ($Z_{ud}$). In order to approximate the optimal solution, we use the time window partitioning method to narrow the gap between $Z_{ov}$ and $Z_{ud}$. This method provides a way to compromise between the optimality and solution time.

In addition, we use a small example to verify the accuracy and rationality of our model. Then we generate our test instances from Solomon's benchmark test samples for VRPTW. We evaluate our solving method by observing the ratio of $Z_{ov}$ and $Z_{ud}$. In the tests without using time window partitioning method, the ratio is all above 0.7, and the average ratio is 0.822. In the tests of time window partitioning method, most instances yield the optimal solution.

## 5.2 Suggestions

1. Since we have constructed a linear mathematical formulation of PDPTW, this formulation can also be modified to deal with dial-a-ride problems. For example, modify the objective function to minimize the customer inconvenience, or to minimize the number of vehicles.

2. Although we use the solution of over-constraint formulation as a hurdle value when solving the under-constraint formulation, the time to find a feasible solution and the lower bound is still lengthy. In fact, the time to find a lower bound can be reduced by some techniques, such as solving the LP relaxation of the under-constrained or original formulation. Notice that the under-constraint method may provide a much tighter lower bound than LP relaxation in some situations.

3. The LP solver we use is LINGO8.0, other commercial optimization packages such as CPLEX, DOT, GAMS, or later version of such kind of software may have shorter solution times for mixed integer programming problems.

4. Small partitioning width can narrow the gap between $Z_{ov}$ and $Z_{ud}$, but also generate larger problem sizes. How to choose a good partitioning width to get the best trade-off would be an issue for further research.

5. The computational experience with the model using LINGO8.0 indicates that problems involving less than 30 sub-jobs can be solved without much difficulty. But it is still troublesome for solving large problems because the problem size expands too fast, especially when using time partitioning method. Further research can focus on the speeding up techniques like branch-and-cut algorithm to deal with large problems.

## References

1. Warren B. Powell, “Optimization models and algorithms: an emerging technology for the motor carrier industry,” IEEE Transactions on Vehicle Technology, vol. 40, No. 1, February 1991.

2. M.W.P Savelsbergh, “Local Search for Routing Problems with Time Windows”, Annals of Operations Research 4, 285-305, (1985).

3. J. Desrosiers (1995), “Time Constrained Routing and Scheduling,” Chapter 2 in M.Ball, T.Magnanti, C.Monma and G..Nemhauser (eds.), Network Routing, Hand Books in Operations Research and Management Science Volume 8, pp.102-116.

4. M. Solomon, “The general Pickup and Delivery problem,” Transportation Science, No. 1, February 1995.

5. F. H. Cullen, J. J. Jarvis and H. D. Ratliff, “Set partitioning based heuristics for interactive routing,” Networks 11, 125-143 (1981).

6. Y. Dumas, J. Desrosiers and F. Soumis, “The pickup and delivery problem with time windows,” European Journal of Operational Research, 54, 7-22(1991).

7. Haibing Li and Andrew Lim, “A metaheuristic for the pickup and delivery problem with time windows”, 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01) , November 2001.

8. William P. Nanry and J. Wesley Barnes, “Solving the pickup and delivery problem with time windows using reactive tabu search,” Transportation Research Part B, 34(2000) 107-121.

9. H. Psaraftis, “A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem,” Transportation Science,

14, 130-154(1980).

10. H. Psaraftis, “Dynamic vehicle problems,” in Vehicle Routing: Methods and Studies, B. L. Golden and A. A. Assad (eds), North-Holland, Amsterdam, 1988.

11. L. F. Frantzeskakis and W. B. Powell, “A successive linear approximation procedure for for stochastic, dynamic vehicle allocation problems,” Transportation Science, 20B, 243-257(1990)

12. Hoong Chuin Lau and Zhe Liang, “Pickup and delivery with time windows: algorithms and test case generation,” 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01) , November 2001.

13. Stefan Irnich, “A multi-depot pickup and delivery problem with a single hub and heterogeneous vehicles,” European Journal of Operational Research, 122(2000) 310-328.

14. H. Onal, B. M. Jaramillo and M. A. Mazzocco, ”Two formulations of the vehicle routing problem: an empirical application and computational experience,” Logistics and Transportation Review, January 1996.

15. Xiubin Wang and Amelia C. Regan, “Local truck load pickup and delivery with hard time window constraints,” Transportation Research Part B, 36(2002) 97-112.

16. M.M.Solomon, “Algorithms for the vehicle routing and scheduling problem with time window constraints”, Operations Research, 41, 469-488, (1987).