

Dynamic Optimization of Cargo Movement by Trucks in Metropolitan Areas with Adjacent Ports

Principal Investigators:

Petros Ioannou

University of Southern California
Center for Advanced Transportation Technologies
Electrical Engineering - Systems, EEB 200B
Los Angeles, CA 90089-2562

Anastasios Chassiakos

California State University at Long Beach
College of Engineering,
Long Beach, CA 90840-5602

Research Group:

Hossein Jula

University of Southern California
Center for Advanced Transportation Technologies
Electrical Engineering - Systems, EEB 324
Los Angeles, CA 90089-2563

Ricardo Unglaub

California State University at Long Beach
College of Engineering,
Long Beach, CA 90840-5602

Final Report,
June 2002



Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, and California Department of Transportation in the interest of information exchange. The U.S. Government and California Department of Transportation assume no liability for the contents or use thereof. The contents do not necessarily reflect the official views or policies of the State of California or the Department of Transportation. This report does not constitute a standard, specification, or regulation.

Abstract

Today, in the trucking industry, dispatchers perform the tasks of cargo assignment, and driver scheduling. The growing number of containers processed at marine centers and the increasing traffic congestion in metropolitan areas adjacent to marine ports, necessitates the investigation of more efficient and reliable ways to handle the increasing cargo traffic. In this report, it is shown that the problem of container movement by trucks can be modeled as a “multi-Traveling Salesmen Problems with Time Windows” (m-TSPTW). A two-phase exact algorithm based on dynamic programming is proposed that will find the best routes for a fleet of trucks. Since the m-TSPTW problem is Nondeterministic Polynomial (NP) hard, the computational time for large size problems becomes very high. For the case of medium to large size problems, we develop two computationally feasible methods: 1) a hybrid methodology consisting of dynamic programming in conjunction with genetic algorithms, and 2) a heuristic insertion method. Furthermore, since the cargo movement in a traffic network is a dynamic problem, we use the heuristic insertion method to add newly arriving customers to the set of customers with advanced requests. Computational results demonstrate the efficiency of the hybrid method for static problems and the insertion method for the dynamic ones.

Table of Contents

DISCLAIMER	I	
ABSTRACT.....	II	Deleted: II
DISCLOSURE	VI	Deleted: VI
ACKNOWLEDGMENTS	VII	Deleted: VII
1 INTRODUCTION.....	1	
2 CARGO MOVEMENT: PROBLEM DESCRIPTION AND FORMULATION.....	4	Deleted: 4
2.1 CONTAINER MOVEMENT	4	Deleted: 4
2.2 PROBLEM DESCRIPTION.....	5	Deleted: 5
2.3 PROBLEM FORMULATION	5	Deleted: 5
3 VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS.....	10	Deleted: 10
3.1 VRPTW	10	Deleted: 10
3.2 MATHEMATICAL FORMULATION FOR VRPTW	10	Deleted: 10
3.3 SPECIAL CASES.....	12	Deleted: 12
3.4 SOLUTION METHODS FOR VRPTW	12	Deleted: 12
3.5 DYNAMIC VRPTW.....	13	Deleted: 12
4 PROPOSED METHODS FOR M-TSPTW	15	Deleted: 13
4.1 SOLUTION METHODS FOR TSPTW AND M-TSPTW.....	15	Deleted: 15
4.2 PROPOSED EXACT METHOD FOR M-TSPTW.....	16	Deleted: 15
4.2.1 <i>Methodology:</i>	16	Deleted: 16
4.2.2 <i>Computational Experiments:</i>	19	Deleted: 16
4.3 GENETIC ALGORITHMS FOR M-TSPTW	19	Deleted: 16
4.3.1 <i>Methodology:</i>	20	Deleted: 19
4.3.2 <i>Computational Experiments:</i>	22	Deleted: 19
4.4 INSERTION METHOD.....	23	Deleted: 19
4.4.1 <i>Methodology:</i>	23	Deleted: 20
4.4.2 <i>Computational Experiments:</i>	26	Deleted: 22
4.5 REAL DATA.....	26	Deleted: 23
4.6 DYNAMIC M-TSPTW	28	Deleted: 23
5 CONCLUSIONS AND RECOMMENDATIONS	31	Deleted: 23
6 IMPLEMENTATION.....	32	Deleted: 26
REFERENCES.....	33	Deleted: 26
		Deleted: 28
		Deleted: 31
		Deleted: 32
		Deleted: 33

List of Figures

Figure 1: Typical routes starting from depot and ending at the same depot. The large empty circle denotes the depot. Each small black circle denotes the origin (O) or the destination (D) of a cargo.....	6	Deleted: 6
Figure 2: Each origin-destination pair in Figure 1 can be grouped as a node.	8	Deleted: 8
Figure 3: Time window at origin $[a_O, b_O]$, destination $[a_D, b_D]$, and time window at destination shifted back in time $[a'_D, b'_D]$	8	Deleted: 8
Figure 4: All possible situations between time window at origin, and time window at destination shifted back in time. The dashed area presents the time window at node OD.....	9	Deleted: 9
Figure 5: A graphical representation of a typical route.	23	Deleted: 23
Figure 6: The graphical representation of waiting time, excess time, service time, and time windows at each node in a typical route.....	24	Deleted: 24
Figure 7: Route cost vs. the degree of dynamisms; $N_T=10$	29	Deleted: 29
Figure 8: Route cost vs. the degree of dynamisms; $N_T=20$	29	Deleted: 29
Figure 9: Route cost vs. the degree of dynamisms; $N_T=50$	30	Deleted: 30

List of Tables:

<i>Table 1. The exact method: computational experiments.....</i>	19	Deleted: 19
<i>Table 2. The hybrid GA method: computational experiments.....</i>	22	Deleted: 22
<i>Table 3. Comparing exact, hybrid GA and insertion methods.....</i>	26	Deleted: 26
<i>Table 4. Real Data solutions Comparison.....</i>	27	Deleted: 27
<i>Table 5. The result of applying appointment system on acquired Real Data.....</i>	28	Deleted: 28

Disclosure

Project was funded in entirety under this contract to California Department of Transportation.

Acknowledgments

We would like to express our sincere gratitude to Dr. Maged Dessouky, Professor at the Department of Industrial and System Engineering at the University of Southern California, for numerous discussions, suggestions and guidance.

We would also like to thank Ms. Patty Senecal, Mr. Mike Johnson, and Mr. J.R. Barba of Transport Express for helping us acquiring real data, and supplying us with useful information.

1 Introduction

Today, the elimination of international trade barriers, lower tariffs and shifting centers of global manufacturing and consumption leads to new dynamics in intermodal shipping. Worldwide container trade is growing at a 9.5% annual rate, while the rate of growth for the U.S. is around 6% (Vickerman 1998). The continuous increase in worldwide container trade have forced terminals to cope with "avalanches" of containers, in such a way that every major port is expected to double and possibly triple its cargo by year 2020 (Taleb-Ibrahimi *et al.* 1993, Ryan 1998).

The growth in the number of containers has already introduced congestion and threatened the accessibility to many terminals. The congestion at a port, in turn, magnifies the congestion in the adjacent metropolitan traffic network and affects the trucking industry on three major service dimensions: travel time, reliability, and cost. Trucking is a commercial activity, and trucking operations are driven by the need to satisfy customer demands and the need to operate at the lowest possible cost (Meyer 1996). This industry is highly competitive, with easy entry into almost any market, with relatively little differentiation between operators and slim profit margins.

Today, in the trucking industry, human operators (dispatchers) still play the major role in cargo assignment, route planning and driver scheduling. Dispatchers inform drivers about traffic conditions, in addition to assisting them in departure/arrival decisions and providing navigational information (Ng *et al.* 1995). Dispatchers currently obtain information about traffic conditions, mostly through radio traffic reports and through information relayed back by the drivers (Hall and Intihar 1997). The growing number of containers at marine centers and the increasing traffic congestion in metropolitan areas necessitates the investigation of more efficient, reliable and systematic ways to handle the increasing amount of cargo in a metropolitan traffic network.

The purpose of this report is to investigate methods for improving the scheduling of trucks, where ISO containers¹ need to be transferred between marine terminals, intermodal facilities, and end customers. The objective is to reduce empty miles, and to improve customer service. As a consequence of reduced miles and better service, container terminals can become more competitive, vehicle emissions will be reduced, and drivers will incur less congestion related delays.

In this report, we show that the container movements by trucks in metropolitan area can be modeled as a multi-Traveling Salesmen Problem with Time Windows (m-TSPTW). This problem is often referred to as the full-truck-load problem in the academic literature (Savelsbergh and Sol 1995). The problem entails the determination of routes for the fleet of trucks so that the total distribution costs are minimized while various requirements (constraints) are met. The m-TSPTW is an interesting special case of the Vehicle Routing

¹ Most containers are sized according to International Standards Organization (ISO). Based on ISO, containers are described in terms of TEU (Twenty-foot Equivalent Units) in order to facilitate comparison of one container system with another. A TEU is 8 feet wide, 8 feet high and 20 feet long. An FEU is an eight-foot high forty-foot container and is equivalent to two TEUs.

Problem with Time Windows (VRPTW) where the capacity constraints are relaxed. Savelsbergh (1985) has shown that finding a feasible solution to the single Traveling Salesman Problem with Time Windows (TSPTW) is a Nondeterministic Polynomial (NP)-complete problem.

Although there has been a significant amount of research on the VRPTW (e.g., see Bodin *et al.* 1983, Golden and Assad 1988, Desrochers *et al.* 1992, Fisher 1995, and Desrosiers *et al.* 1995), there has been little work on the m-TSPTW. Since the m-TSPTW is a relaxation of the VRPTW, it may appear at first that the procedures developed for the latter could be applied to the m-TSPTW. However, as Dumas *et al.* (1995) point out, these procedures are not well suited to the m-TSPTW. Hence, new procedures need to be developed for the m-TSPTW. We note that Dumas *et al.* (1995) presented an efficient exact solution procedure for the single vehicle TSPTW. However, in contrast to the simple transformation of the m-TSP to the TSP, m-TSPTW cannot be easily converted to a single vehicle TSPTW.

In this report, we propose three methodologies for solving the m-TSPTW problem.

- An exact method based on Dynamic Programming (DP) is proposed. The method consists of two phases: 1) generating feasible solutions, and 2) finding the optimum solution among all feasible solutions (set-covering problem). Computational experiments show that the proposed exact method is efficient for relatively small size problems consisting of few nodes (up to 15 nodes with maximum window size 2 hours).
- For medium size problems, we develop a hybrid methodology consisting of dynamic programming for generating feasible solution in conjunction with genetic algorithms (GA). The GA algorithm is used to find a ‘good’ solution among all feasible solutions. Experimental results show the efficiency of GA set-covering algorithm for medium size problems (about 50 nodes with maximum window size less than 2 hours).
- A heuristic insertion method is developed for large size problems (more than 50 nodes). In this method, nodes are sequentially inserted into already determined routes. A new route is generated whenever a node cannot be added to any existing route. Experimental results show that the method is computationally very fast and it is fairly efficient for medium to large size problems (more than 50 nodes with very wide window size, e.g. 9 hours).

It is worth mentioning that most techniques and models used in transportation planning, scheduling, and routing use ‘known’ static data as their input (e.g., see Golden and Assad 1988, Desrochers *et al.* 1988, Laporte 1992, Savelsbergh and Sol 1995). In the real world, however, truck operations contain a fairly high degree of uncertainties (e.g., see Powell *et al.* 1995, Powell 1996). Since cargo assignment and route planning is a dynamic problem, any efficient algorithm should also be *dynamic*. Despite this fact, little work has been done in this area (Psaraftis 1995). For a dynamic system, two methodologies could be employed: 1) insertion methods and 2) adaptation of static algorithms. The results of our computational experiments, presented in Section 4, indicate that only the insertion method is well suited to be considered as a dynamic algorithm for real time systems. The

insertion method is very fast and it is fairly efficient especially for the problems with large number of customers.

In this report, we assume a deterministic transportation network. We assume that the travel times on the links of the network are independent of the time when vehicles enter that links. These assumptions are made to simplify the problem, get better insight, make it possible to model the problem mathematically and come up with a close analytical solution. In our future work, we intend to extend the following study into the non-stationary stochastic networks.

The report is organized as follows. In Section 2, the container movement and trucking operations in metropolitan areas are described, and is formulated as an asymmetric m-TSPTW. In Section 3, we study a more general class of m-TSPTW, the VRPTW problem, where solution methods for VRPTW are briefly reviewed, and dynamic aspects of VRPTW are discussed in detail. In Section 4, we show that the VRPTW solution methods are not well suited for m-TSPTW problem, and propose three methodologies for solving the m-TSPTW problem.

2 Cargo Movement: Problem Description and Formulation

In this section, it is shown that the container movements by trucks can be modeled as an asymmetric multi-Traveling Salesmen Problem with Time Windows (m-TSPTW). We start with describing the container movement and trucking operations in metropolitan areas.

2.1 Container Movement

Today, in trucking industry, human operators (dispatchers) perform the tasks of cargo assigning, route planning and driver scheduling. Each day, the list of cargoes to be handled during the day is passed to the dispatcher early in the morning. For containers, this list contains the information about the origin and the destination of containers, if they are over weight, contain HAZMAT², or need to be handled before any specific time. The dispatcher will assign a driver to each container based on the availability of the driver and his skills. At most two containers will be assigned to each driver at a time. The dispatcher will assign two containers if the delivery point of the first one is relatively close to the pick up location of the second one. Otherwise, only one container is assigned to the driver. Upon finishing his job(s), the driver would ask the dispatcher for new jobs. If new jobs need to be accompanied by formal documentation or no job is available at the time, the driver will be asked to go back to the depot, otherwise the new job(s) will be given to him via the cell phone.

In today's container terminals two operations are taking place: wheeled, and ground (Ioannou *et al.* 2000). A wheeled operation is one in which the container is brought into the container yard on a chassis, and then brought to the ship on the same chassis, and lifted off. This empty chassis can then be used for inbound containers. The wheeled operations, where possible, are more efficient than the grounded ones in terms of loading and unloading rates as they require no container transfer and are random access. That explains why the trucking companies are in favor of terminals with wheeled operations. Ground operations, on the other hand, are very efficient in terms of space, since the containers can be stacked 4-5 high when full, and even higher when empty. Ground operations require more effort for storage and retrieval, since the container must be transferred via lifting equipment, and often, multiple lifts are required in the yard where containers are stacked on top of one another. It takes much longer for the trucker to be served, since he/she has to get a chassis at the terminal, find the container stack location, wait for his/her turn for the equipment to lift the container from the stack and put the container on the chassis.

In today's container industry, there are a lot of discussions about appointment windows. The appointment systems are being considered as part of a solution for terminal congestion. These systems are coming about because of the terminals' need to 'manage

² HAZardous MATerial.

the demand' (the flow of trucks). For instance, the Hanjin terminal at Port of Long Beach has just started informing the local trucking companies of the list of containers they want to be picked up on the hoot shift (i.e., 3:00 a.m. - 7:00 a.m.). However, no standard has yet been maintained of how the appointment systems should be implemented.

As it was noted before, trucking operations are driven by the need to satisfy customer demands and the need to operate at the lowest possible cost. These facts magnify the need for finding better ways of performing trucking operations in metropolitan areas adjacent to the ports.

2.2 Problem Description

The problem of interest, described above, can be stated more formally as follows: A set of loads (containers) needs to be moved in a metropolitan (local) area. The local area contains one truck depot (which thereafter will be called depot), as well as many end customers including marine terminals and intermodal facilities. Associated to each load is hard time windows imposed by customers for pickup and delivery at origin and destination points, respectively.

A set of trucks (vehicles) is deployed to move the loads among the customers, and the depot. Each truck can only serve a single load (e.g. one FEU³ size container) at a time, and initially, all trucks are located at the depot. We assume that each driver may not be at the wheel for more than a certain number of hours (working shift) in each working day and has to drive his truck back to the depot within this time limit.

The objective is to minimize the total cost of providing service to loads within their specified time constraints.

2.3 Problem Formulation

Let L be a set of n cargos (loads) l_i to be transferred in a transportation network G , i.e. $L = \{l_1, l_2, \dots, l_n\}$; and V be a set of p vehicles labeled $v_i, i = 1, 2, \dots, p$ assigned to transfer the cargos, i.e. $V = \{v_1, v_2, \dots, v_p\}$.

We assume that, at any time, a vehicle $v_i \in V$ can transfer at most a single cargo, say $l_i \in L$, and that the information of the origin and the destination of the cargo l_i is known in advance. We denote by $O(l_i)$ and $D(l_i)$ the origin and the destination of the cargo l_i , respectively. The cargo l_i shall be picked up from its corresponding origin during a specific period of time known as pickup time window and denoted by $[a_{O(l_i)}, b_{O(l_i)}]$, where $a_{O(l_i)} < b_{O(l_i)}$. Likewise, cargo l_i shall be delivered at its corresponding destination during a delivery time window denoted by $[a_{D(l_i)}, b_{D(l_i)}]$, where $a_{D(l_i)} < b_{D(l_i)}$.

³ FEU: Forty-foot Equivalent Units (See also footnote 1 on page 1).

Let's assume that to the vehicle $v_i \in V$ the total number of $K(i)$ cargos are assigned to be transferred. Let $\delta_{ik} \in L$ be the k th cargo assigned to vehicle v_i among these cargoes. The sequence of cargos assigned to the vehicle v_i is called route and is denoted by r_i , i.e. $r_i = \{\delta_{i1}, \delta_{i2}, \dots, \delta_{ik}, \dots, \delta_{iK(i)}\}$. Route r_i is said to be feasible if it satisfies the time window constraints at the origins and the destinations of all assigned cargoes, and the total time needed for traveling on the route is less than a certain amount of time called the working shift (time) and denoted by T .

It should be noted that:

$$\bigcup_{i=1}^p r_i = L, \text{ and } r_i \cap r_j = \emptyset, \text{ for } \forall i, j = 1, \dots, p \text{ and } i \neq j, \text{ where } \emptyset \text{ indicates the empty set.}$$

Figure 1 shows typical three feasible routes (r_1 , r_2 , and r_3) starting from depot and ending at the same depot. Solid lines, in Figure 1, illustrate the traveling between origin and destination when the vehicle is loaded, while dashed lines indicate empty traveling between destination and origin.

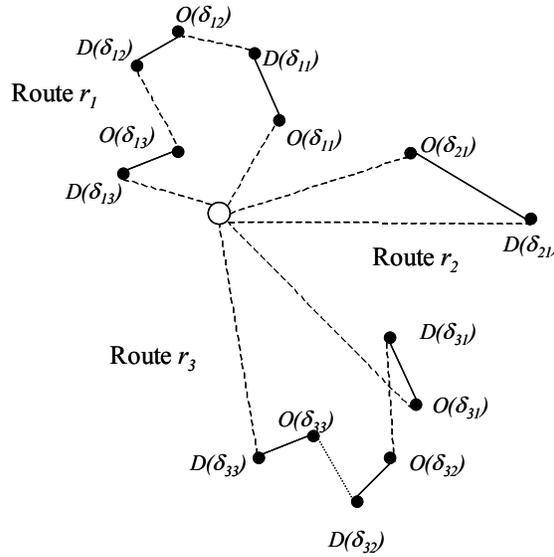


Figure 1: Typical routes starting from depot and ending at the same depot. The large empty circle denotes the depot. Each small black circle denotes the origin (O) or the destination (D) of a cargo.

Let's denote by $f(r_i)$ the cost associated with each route i . $f(r_i)$ can be formulated as follows:

$$f(r_i) = \sum_{k=1}^{K(i)} c_{O(\delta_{ik})D(\delta_{ik})} + \sum_{k=0}^{K(i)} c_{D(\delta_{ik})O(\delta_{ik+1})} \quad (1)$$

Where:

$c_{O(\delta_{ik})D(\delta_{ik})}$ is the cost of carrying the k th cargo from its origin to its destination, (solid lines between each pair of $O(\delta_{ik})$ and $D(\delta_{ik})$ in Figure 1),

$c_{D(\delta_{ik})O(\delta_{ik+1})}$ is the cost of empty traveling between the destination of the k th cargo to the origin of the $(k+1)$ th cargo, (dashed lines between each pair of $D(\delta_{ik})$ and $O(\delta_{ik+1})$ in Figure 1). Depot is denoted by $k=0$ and $k=n+1$, and

$K(i)$ is the total number of cargos handled in route r_i by vehicle v_i .

The objective is to find optimum routes for p vehicles providing the services to n cargos by traveling between origins and destinations of cargos and satisfying the time window constraints such that the completion of handling all cargos results in minimizing the total travel cost. The objective function, J can be written as follows:

$$J = \text{Min} \sum_{i=1}^p f(r_i), \quad (2)$$

Let's assume that the travel cost for a vehicle, either loaded or empty, is static, and the cost associated with transferring a cargo $l_i \in L$ between its origin and destination, $c_{O(l_i)D(l_i)}$, is independent of the order of transferring the cargo by a vehicle. Let's also assume that the fleet of vehicles is homogenous. Therefore, no matter what the assignment and order of handling n cargoes are, the costs $c_{O(\cdot)D(\cdot)}$ don't affect the cost function in Equation (2) and can be considered to be zero. That is the total cost function in Equation (2) is only affected by the cost associated with vehicles' empty traveling between the destinations of the k th and $(k+1)$ th cargoes; and the problem of interest is reduced to finding the best feasible assignment and sequencing of n cargoes to p vehicles such that the total empty travel cost of the vehicles is minimum.

In this report, we assume a deterministic transportation network in which the network parameters are known a priori. We also assumed that the travel time on a link of the network is independent of the time when a vehicle enters that link. These assumptions are made to simplify the problem, get better insight, and make it possible to model the problem mathematically and come up with a close analytical solution. In such a network, the order of the pick up/delivery points won't affect the cost of operation.

In our future work, we intend to consider non-stationary stochastic networks in which the travel times on the links as well as the service times at the nodes of the network are stochastic processes. In other words, the travel time between each two nodes of the network is a random variable, which depends on the time when a vehicle enters the link. In such a network, the sequence of the pick up/delivery points will affect the cost of the cargo movement.

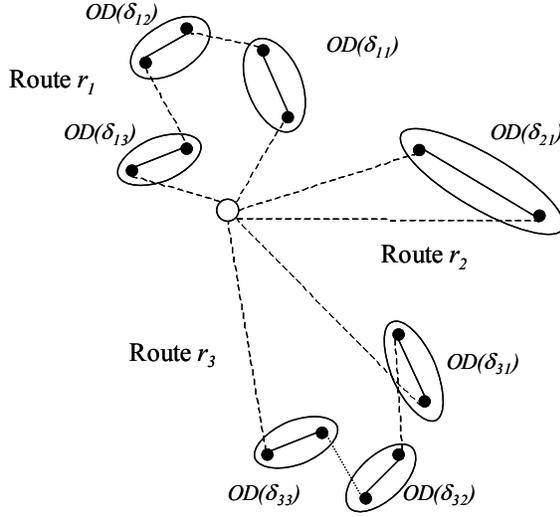


Figure 2: Each origin-destination pair in Figure 1 can be grouped as a node.

Thus, each origin-destination pair, $O(\delta_{ik})-D(\delta_{ik})$, in Figure 1 can be replaced by a node $OD(\delta_{ik})$ where $\delta_{ik} \in r_i$ and $k=1, \dots, K(i)$, and the cost between two nodes is equal to the cost of empty traveling between the destination of the first node to the origin of the second one (see Figure 2).

The time window at node $OD(\delta_{ik})$ can be expressed in terms of: 1) time window at its origin, 2) time window at its destination, and 3) the traveling time between the origin and the destination, $t_{O(\delta_{ik})D(\delta_{ik})}$. Figure 3 demonstrates a typical relation between these three factors and the time window $[a'_D, b'_D]$, where $[a'_D, b'_D]$ is the time window at destination shifted back in time by $t_{O(\delta_{ik})D(\delta_{ik})}$. For the sake of simplicity, we eliminate all subscripts δ_{ik} in Figure 3.

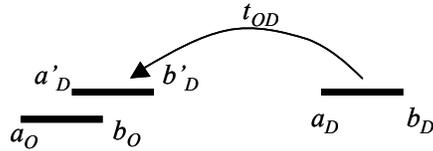


Figure 3: Time window at origin $[a_O, b_O]$, destination $[a_D, b_D]$, and time window at destination shifted back in time $[a'_D, b'_D]$.

Figure 4 presents all possible situations between time windows $[a'_D, b'_D]$ and $[a_O, b_O]$. The dashed areas, in Figure 4, indicate the time window at the origin of node OD during which a vehicle can be loaded and yet to meet the time window constraint at destination. Case IV is infeasible and may not happen in real situation.

The problem of interest can now be restated as follows: p vehicles are initially located at the depot. They have to visit nodes $OD(l_i)$, $i=1,\dots,n$; the task is to select some (or all) of these vehicles and assign routes to them such that each node is visited exactly once during the time window $[a_i, b_i]$, where $[a_i, b_i]$ can be expressed as follows (see Figure 4).

$$a_{l_i} = a_{O(l_i)}$$

$$b_{l_i} = \min(b_{O(l_i)}, b_{D(l_i)} - t_{O(l_i)D(l_i)}) \quad (3)$$

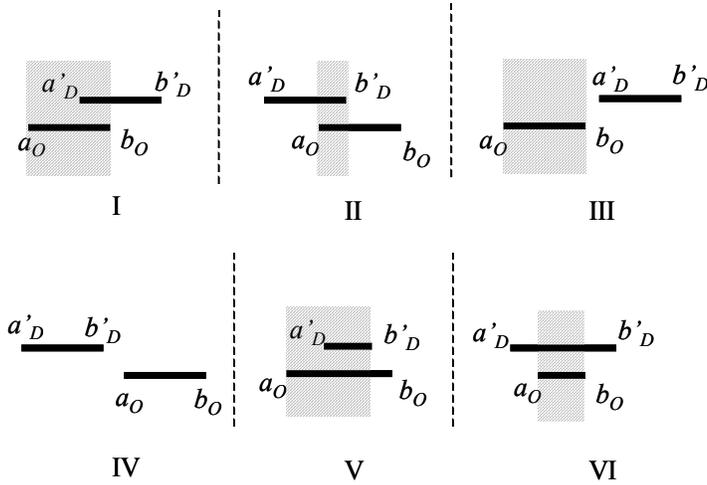


Figure 4: All possible situations between time window at origin, and time window at destination shifted back in time. The dashed area presents the time window at node OD .

The problem now falls in the class of asymmetric Multi-Traveling Salesmen Problems with Time windows (m-TSPTW). In m-TSPTW, m salesmen are located in a city (i.e. node: $n+1$) and have to visit n cities (nodes: $1,\dots,n$). The task is to select some or all of the salesmen and assign tours to them such that in the collection of all tours together the cost (distance) is minimized and each city is visited exactly once within a specified time window (Reinelt 1994). The problem is asymmetric since the traveling cost between each two nodes i and j depends on the direction of the move. Note that,

$$c_{OD(l_i)OD(l_j)} = c_{D(l_i)O(l_j)} \neq c_{D(l_j)O(l_i)} = c_{OD(l_j)OD(l_i)}. \quad (4)$$

3 Vehicle Routing Problems with Time Windows

In Section 2, the movement of containers in metropolitan areas was discussed in detail. It was shown that the problem falls in the class of asymmetric Multi-Traveling Salesmen Problems with Time windows (m-TSPTW). The m-TSPTW problem is an interesting special case of the Vehicle Routing Problem with Time Windows (VRPTW) where the capacity constraints are relaxed. The main purpose of this section is to review the class of VRPTW problems, present its mathematical formulation and discuss the special cases. Solution methods for VRPTW are briefly reviewed, and finally, the dynamic aspects of VRPTW are discussed.

3.1 VRPTW

The Vehicle Routing Problem with Time Windows (VRPTW) is a generalization of the Vehicle Routing Problem (VRP) involving the added complexity of time windows. The VRP and a variety of its practical applications have been the subject of a wide body of research (Bodin *et al.* 1993, Golden and Assad 1988, Laporte 1992, Fisher 1995).

The VRPTW can be defined as follows: A number of vehicles, each with a given capacity, is located at a single depot and must serve a number of geographically dispersed customers. Each customer has a given demand and must be served within a specified time window. The objective is to minimize the total cost of travel (Desrochers *et al.* 1988).

Time windows arise naturally in problems faced by business organizations. Time windows can be hard or soft. With hard time windows, the delivery times cannot be made outside the time window. That is, if a vehicle arrives at a customer location earlier than the beginning of the time window, it has to wait till the customer is ready to begin service. However, the vehicle cannot be served if it arrives at the customer location after the latest time of the time window. In contrast, in the case of soft time windows, the time windows can be violated at a cost (Desrosiers *et al.* 1995). In this report, we are focusing on hard time windows.

3.2 Mathematical formulation for VRPTW

Let $G=(ND,A)$ be a graph with node set $ND=\{o,d,N\}$ and arc set $A=\{(i,j) \mid i,j \in ND\}$. The nodes o and d represent the single depot (origin-depot and destination-depot), and $N=\{1,2,\dots,n\}$ is the set of customers. To each arc $(i,j) \in A$, a cost c_{ij} and a duration of time t_{ij} are associated representing the cost and the time of traveling between nodes i and j , respectively. In addition, to each node $i \in ND$, a service time s_i , a time window $[a_i, b_i]$, and a load variable q_i are associated. The service time s_i is the duration of the time for a vehicle to be served at node i , and a_i and b_i are the earliest and latest time to visit node i , respectively. An arc $(i,j) \in A$ is feasible iff $a_i + s_i + t_{ij} \leq b_j$. Let V be the set of vehicles v with capacity Q . A route in the graph G is defined as assigning a set of nodes $r^v = \{o, w_1^v, w_2^v, \dots, w_k^v, d\}$ to vehicle v such that each arc in r^v belongs to A , the capacity of the

vehicle is not exceeded, and the time service that begins at node $j \in r^v$ is within the time window of that node.

Let's also define:

$x_{ij}^v=1$ if arc $(i,j) \in A$ is traveled by vehicle v and is in the optimal path⁴. $x_{ij}^v=0$ otherwise,

L_i^v is the total load in vehicle v after serving node i , and

T_i^v is the time when service begins at node i by vehicle v .

The VRPTW can be formulated as follows:

$$\text{Min } \sum_{v \in V} \sum_{(i,j) \in A} c_{ij} x_{ij}^v \quad (5a)$$

$$\text{Subject to: } \sum_{v \in V} \sum_{j \in N \cup \{d\}} x_{ij}^v = 1 \quad \forall i \in N \quad (5b)$$

$$\sum_{v \in V} \sum_{j \in N} x_{oj}^v \leq |V| \quad (5c)$$

$$\sum_{j \in N \cup \{d\}} x_{ij}^v - \sum_{j \in N \cup \{o\}} x_{ji}^v = 0 \quad \forall i \in N, v \in V \quad (5d)$$

$$x_{ij}^v (T_i^v + s_i + t_{ij} - T_j^v) \leq 0 \quad \forall i, j \in ND, v \in V \quad (5e)$$

$$a_i \leq T_i^v \leq b_i \quad \forall i \in ND, v \in V \quad (5f)$$

$$x_{ij}^v (L_i^v + q_i - L_j^v) \leq 0 \quad \forall i, j \in ND, v \in V \quad (5g)$$

$$q_i \leq L_i^v \leq Q \quad \forall i \in ND / \{o\}, v \in V \quad (5h)$$

$$L_o^v = 0 \quad \forall v \in V \quad (5i)$$

$$x_{ij}^v \in \{0, 1\} \quad \forall (i,j) \in A, v \in V \quad (5j)$$

Constraints (5b) require that only one vehicle visit each node in N . Constraints (5c) ensure that at most $|V|$ number of vehicles are used. To fix the number of vehicles, the inequality should be replaced by equality. Constraints (5d) guarantee that the number of vehicles leaving node j is the same as the number of vehicles entering the node. Therefore, constraints (5b)-(5d) together enforce that at most $|V|$ number of vehicles visit all nodes in N only once. Constraints (5e) enforce the time feasibility condition on consecutive nodes. Constraints (5f) specify the time window constraints at each node. Constraints (5g)-(5i) guarantee the feasibility of the loads, and finally the constraints (5j) are the binary constraints.

⁴ The optimal path is a cycle of all nodes with the smallest possible total cost of arcs.

3.3 Special Cases

multi-Traveling Salesmen Problem with Time Windows (m-TSPTW): The m-TSPTW emerges in the full-truck-load problem (Savelsbergh and Sol 1995). This problem is our main interest here, since it is motivated by truck service applications found in the local operations where ISO containers need to be transferred between marine terminals, intermodal facilities, and end customers (see Section 2). The m-TSPTW is a special case of the m-VRPTW in which the vehicle capacities are infinite. The mathematical formulation of the problem can be obtained by eliminating constraints (5g)-(5i). In this report, and due to the social constraints imposed on truck drivers, we add another constraint to the standard m-TSPTW formulation as follows:

$$T_d^v - T_o^v \leq T \quad \forall v \in V \quad (5k)$$

Constraint (5k) requires that each vehicle shall be used less than a certain number of hours per day.

Traveling Salesman Problem With Time Windows (TSPTW): The special case of a m-TSPTW in which the number of vehicles is limited to one is TSPTW. That is, we replace constraint (5c) with an equality and set $|V|=1$.

Multi-tour Traveling Salesman Problem (m-TSP): If we relax constraints (5e) and (5f), the problem will be equivalent to an m-TSP. Furthermore, if the number of vehicles is limited to one, the problem is equivalent to a Traveling Salesman Problem (TSP). TSP is the most prominent member of the rich set of combinatorial optimization problems known to be NP-hard. An excellent survey on computational solutions for the TSP is presented in Reinelt (1994). An m-TSP can be easily transformed to a TSP involving only one salesman (see Bellmore and Hong 1974). However, it should be noted that in contrast to the relation between the TSP and the m-TSP, the m-TSPTW cannot be transformed into an equivalent TSPTW.

Vehicle Scheduling Problem (VSP): If the time windows in (5f) consist of a single value, in other words, if $a_i=b_i \quad \forall i \in N$, the m-TSPTW becomes VSP. It should be noted that VSP is solvable as the Minimum Cost Flow Problem (MCFP) in polynomial time (Lee 1992) and, thus, it is not NP hard. This class of problems is very practical and arises in many fields, such as airline, rail, school bus, and urban transportation.

3.4 Solution methods for VRPTW

The solution methods for VRPTW can be grouped into the following major categories: exact methods, and heuristic methods.

Exact methods: The first paper proposing the exact method for solving VRPTW was published in 1987 (Kolen *et al.* 1987). Since then a number of papers have been published and the algorithms almost all use one of the following three principles: dynamic programming— (Kolen *et al.* 1987), Lagrange relaxation (Fisher 1994), and Dantzig-Wolfe decomposition/column generation (Desrochers 1992).

Heuristic methods: Heuristic methods for VRPTW have been studied since 1970's. These methods are mainly based on two phases: 1) route construction, and 2) route improvement. A good review and comparison of early heuristics can be found in Solomon (1987) and in Desrosiers *et al.* (1995). Recently, new approaches based on modern heuristics and meta-heuristics, including tabu search (TS), genetic algorithms (GA) and simulated annealing (SA), are being applied to VRPTW (e.g. see Taillard and Badeau 1997). Savelsbergh (1985) has shown that finding a feasible solution to the VRPTW is an NP-complete problem. Consequently, exact methods are efficient for small size problems. Larger size problems, however, can only be solved using heuristic methods.

3.5 Dynamic VRPTW

Most techniques and models used in transportation planning, scheduling, and routing including those used for VRPTW, use 'known' static data as their input. For instance, it is assumed in most VRPTW work that the customer demands, travel costs, and travel times are static and fully known in advance. In the real world, however, and especially in the trucking industry, truck operations contain a fairly high degree of uncertainties. The traffic characteristics of the selected routes may change (i.e. due to congestion in the route) during the truck operations. Unexpected delays at customer locations may happen. The customer may also alter the requested time window for service after the set of routes has been determined for trucks. Furthermore, at the time of cargo assignment and route planning, carriers may know only a portion of cargoes to be served. That is, new cargoes may be added/extracted to/from the set of the tasks in an unpredictable fashion.

The VRP is *dynamic* if information (input) on the problem is made known to the decision-maker or is updated concurrently with the determination of the set of routes (Psaraftis 1995). The dynamic VRP has recently emerged as an active area of research due to recent technological advances in information and communication technologies (Psaraftis 1995, Powell *et al.* 1995, Gendreau and Potvin 1997). These technologies such as Electronic Data Interchange (EDI), Global Positioning Systems (GPS), and Geographical Information Systems (GIS) significantly enhance one's ability in acquiring data in real time. However, technology alone is not sufficient to ensure success. The need remains to develop *dynamic*, quick and efficient algorithms capable of providing good solutions in real time (Crainic and Laporte 1997).

Although the dynamic methods and models undoubtedly present the 'wave of the future' (Powell *et al.* 1995), little work has been done in this area. Perhaps the dynamic Dial-A-Ride Problem (DARP)⁵ is among the most researched problems in transportation planning (Gendreau and Potvin 1997). In this problem, a new customer's request is added

⁵ DARP is a special case of Pickup and Delivery Problem with Time Windows (PDPTW). In PDPTW, there is a single depot, a number of vehicles with given capacities, and a number of customers with given demands. Each customer must be served to pick up goods at his origin during a specified time window, and to deliver the items at his destination during another specified time window. The objective is to minimize the total travel cost. DARP arises in transportation systems for handicapped and the elderly. In these situations, the temporal constraints imposed by the customers strongly restrict the total vehicle load at any instant of time, and the capacity constraints are of secondary importance.

to the list of tasks, or is refused service. Two approaches have been reported in the literature: 1) insertion methods and 2) adaptation of static algorithms. In insertion methods, a fair amount of requests are known beforehand and are scheduled statically. Then, real time requests are incorporated (inserted) in the initial solution framework (Wilson and Colvin 1977, Jaw *et al.* 1986, Madsen *et al.* 1995). In the second method, the dynamic problem is solved as a sequence of static problems. Psarsftis (1980, 1988) used a Dynamic Programming (DP) algorithm initially to solve a single vehicle, static DARP. The DP algorithm is then applied repetitively when new requests occur.

4 Proposed Methods for m-TSPTW

As it was stated in Section 2, the m-TSPTW is a special case of the Vehicle Routing Problem with Time Windows (VRPTW) where the capacity constraints are relaxed. Consequently, one may think of applying the same solution methods for m-TSPTW by relaxing the capacity constraints in VRPTW. Although the idea of using solution methods on VRPTW in m-TSPTW looks very rational, the experimental results show otherwise. In their work, Dumas *et al.* (1995) state: ‘*Even though the TSPTW is a special case of VRPTW, the best known approach to the latter problem (Desrochers et al. 1992) is not well suited to solve the TSPTW. This column generation approach would experience extreme degeneracy difficulties in this case.*’

Since solution methods for VRPTW are not well suited for m-TSPTW, researchers have sought methods tailored for the TSPTW and m-TSPTW. In this Section, first we review the existing solution methods for TSPTW and m-TSPTW. Despite the importance of m-TSPTW in trucking industry, the research on TSPTW and m-TSPTW has been scant (Desrosiers *et al.* 1995). In this report, three methodologies for solving m-TSPTW are proposed. Experimental results are used to show the efficiency of each method regarding the optimality of the solution and the problem size.

4.1 Solution Methods for TSPTW and m-TSPTW

Since finding a feasible solution to TSPTW and m-TSPTW is NP-complete problem (Savelsbergh 1985), most research has been focused on heuristic algorithms. Lin and Kernighan (1973) proposed a heuristic algorithm based on k -interchange concept for TSPTW. The method involves the replacement of k arcs currently in the solution with k other arcs. Lee (1992) developed two heuristics based on the Vehicle Scheduling Problem (VSP) for the m-TSPTW. The VSP algorithms are exact in that they can find the optimal solution to the VSP in polynomial time. However, solutions found by VSP algorithms may be infeasible for the m-TSPTW. Two construction heuristics are developed to assign each customer to a route. Improvement heuristics are then developed to combine the initial routes. Calvo (2000) proposed using a new heuristic method for the TSPTW based on solving an auxiliary assignment problem. To find better solutions, the algorithm used two objective functions. As soon as it was trapped in a local minimum, the algorithm used the second objective function to widen its neighborhood region.

Despite the difficulty of the TSPTW, few authors have focused on exact solution. Dumas *et al.* (1995) used Dynamic Programming (DP) enhanced by a variety of elimination tests to solve the single vehicle TSPTW optimally. These tests took advantage of the time window constraints to significantly reduce the number of arcs in the graph to eliminate states. The authors managed to solve problems of up to 200 nodes with small window size, and problems of up to 80 nodes with larger time windows.

As we noted previously, in contrast to the relation between the TSP and the m-TSP (see Reinelt 1994), the m-TSPTW cannot be transformed into an equivalent TSPTW. In this

report, the following methodologies are developed to extend the earlier work on the single vehicle TSPTW:

- 1) An exact two-phase Dynamic Programming (DP),
- 2) A hybrid methodology consisting of DP in conjunction with genetic algorithms (GA), and
- 3) A heuristic insertion method.

We next describe our approaches for solving the m-TSPTW.

4.2 Proposed Exact Method for m-TSPTW

A two-phase Dynamic Programming (DP) based method is proposed for solving m-TSPTW. The method is the extension of the algorithm used for TSPTW and proposed in (Desrosiers1995 and Dumas 1995). In phase one, a forward DP is used to generate all feasible solutions, which will be called states. To reduce the computational time and the number of states, extensive elimination tests are performed before and during running the DP algorithm. The elimination tests take advantage of the time window constraints in (5e), (5f) and (5k) to significantly reduce the number of states. The outcome of the first phase will be sets of feasible solutions. The sets, then, will be fed to another DP algorithm in order to find a set of routes with minimum total cost which covers all nodes. That is, the second DP solves a set-covering problem in order to extract the optimum set of solutions among all sets of solutions. In the followings, we explain the exact method in detail.

4.2.1 Methodology:

Phase One: In this Section, for convenience, we adopt the same notation as in Subsection 3.2. We require that the triangular inequality hold for both the travel costs and the travel times between each two nodes of the Graph G (See Section 3 for definition of G). That is, for any $i, j, k \in ND$, we have $c_{ik} \leq c_{ij} + c_{jk}$ and $t_{ik} \leq t_{ij} + t_{jk}$.

We define the state (S, i, t) as follows: $S \subseteq N$ is an unordered set of visited nodes, $i \in S$ is the last visited node, and t is the time service begins at node i . Assigned to each state is a cost denoted by $F(S, i, t)$ and defined as the least cost with minimum spanned time of a path starting at node $\{o\} \in ND$ passing through every node of S exactly once and ending at node i . Note that, there are several paths that visit set S and end at node i . Among them, we choose the one with minimum cost and minimum spanned time (see the state elimination test ‘II.a’, below).

Let $fr(S) \in N$ be the first node in set S . The starting time of set S is denoted by $t_{fr(S)}$ and is defined as $t_{fr(S)} = \max(a_o, a_{fr} - s_o - t_{o,fr})$. This time indicates the earliest possible vehicle dispatching time from the depot such that no waiting time is necessary at $fr(S)$.

In order to reduce the computational time, two types of elimination tests are performed: arc elimination, and state elimination.

I) Arc elimination tests: The arc elimination tests take advantage of the time window constraints (5e), (5f) and (5k) to significantly reduce the number of states. These tests are performed before and during running the DP algorithm.

a) Arc elimination before running the algorithm. Let $EAT(i,j)$ be the earliest arrival time at node j from node i . $EAT(i,j)$ defined as follows:

$$EAT(i,j)=a_i+s_i+t_{ij} \quad (6)$$

Let also $BEFORE(j)$ be the set of all nodes that should be visited before node j , and is defined as follows:

$$BEFORE(j)=\{k \in ND \mid EAT(j,k) > b_k\} \quad (7)$$

Nodes which can not be covered by set $BEFORE(j)$ will be added to set $FORBID(j)$. In the arc elimination tests, as soon as the algorithm reaches a node j which cannot be added to set S of state (S,i,t) , neither node j nor $FORBID(j)$ will be explored at any other state generated from that state. The Forbidden nodes for set S will be kept in the set $U(S)$.

b) Arc elimination during running the algorithm. Given state (S,i,t) , $a_i \leq t \leq b_i$, the time to visit node $j \in N$ after node i is $t+s_i+t_{ij}$. The reachability time of depot, $\{d\} \in ND$, after visiting node j is denoted by t' and is defined as $t' = \max(a_j, t+s_i+t_{ij})+s_j+t_{j,d}$. Node j can be added to set S if all of the following tests are satisfied.

$$t+s_i+t_{ij} \leq b_j \quad (8a)$$

$$t' \leq b_d \quad (8b)$$

$$t'-t_{\hat{r}(S)} \leq T \quad (8c)$$

Note that, if node j can not meet any of the tests in (8), node j as well as all nodes in $FORBID(j)$ will be kept in the set $U(S)$ and will not be explored at any other states generated from S .

II) State elimination tests. This set of tests implement the Dynamic Programming algorithm to reduce the number of states: a) during performing phase one, and b) after finishing this phase.

a) State elimination during running the algorithm. Given states (S,i,t_1) and (S,i,t_2) , the second state is eliminated if $t_1 \leq t_2$ and $F(S,i,t_1) \leq F(S,i,t_2)$.

b) State elimination after finishing the algorithm, Given states (S,d,t_1) and (S,d,t_2) , the second state is eliminated if $F(S,d,t_1) \leq F(S,d,t_2)$. Test II-b reduces the number of states passing to the next phase in order to reduce the computational time in phase II.

Algorithm:

Step 1: Initialization: level $l=1$,

$$\text{Form } \{S,i,t\}: S=\{o,i\}, t=t_{\hat{r}(S)}, F(S,i,t)=c_{o,i},$$

$$U(S)=FORBID(i) \quad \forall (o,i) \in A,$$

Step 2: During: Set level $l=l+1$

For all states S in $l-1$

If $U(S) \cup \mathcal{S} \neq \mathcal{N}$,

For $\forall (i,j) \in A, j \notin U(S)$, i is the last node in S , j satisfies tests in (8)

Generate new state $\{S_j, j, t_j\}$

$S \cup \{d\} \cup \{j\} \rightarrow S_j$,

$t_j = \max(a_j, t + s_i + t_{ij})$

$F(S_j, j, t_j) = F(S, i, t) + c_{ij}$

$U(S_j) = U(S) \cup \text{FORBID}(j)$

Perform test II-a

Step 3: Termination: If no node was generated in level l

Form $S \cup \{d\}$, $F(S, d, t') = F(S, j, t) + c_{j,d}$ where j is the last node in S

Perform test II-b

Stop

Otherwise: Go to Step 2

Phase Two: The outcome of the first phase is sets of feasible solutions (routes) with their associated costs. Let's assume that R routes were generated in Phase I. Let x_r be one if the route $r \in R$ is selected among the optimum routes, and zero otherwise. Associated to route r is the cost F_r which was calculated in the previous phase. Let also β_{ir} be equal to one if route r visits node $i \in \mathcal{N}$, and zero otherwise. The set-covering problem can be formulated as follows:

$$\text{Min} \quad \sum_{r=1}^R F_r x_r \quad (9a)$$

Subject to:

$$\sum_{r=1}^R \beta_{ir} x_r = 1 \quad (9b)$$

$$x_r = \{0,1\} \quad (9c)$$

In order to find the set of routes in (9) which covers all nodes in G exactly once with minimum total cost, the routes generated in Phase I are fed to another DP algorithm. That is, the second DP solves a set-covering problem in order to extract the optimum set of solutions among all sets of solutions. It should be mentioned that the set covering problem has also been proven to be NP-hard (Garey 1979).

In phase II, the state (St, ν) is defined as follows: $St \subseteq \mathcal{N}$ is an unordered set of visited nodes, and ν is the number of routes (vehicles) forming the state St . Assigned to each state is a cost denoted by $g(St, \nu)$ and is defined as the least total costs of routes forming

St (covering every node of St). Given states (St, v_1) and (St, v_2) , the second state is eliminated if $g(St, v_1) < g(St, v_2)$. Furthermore, if $g(St, v_1) = g(St, v_2)$ the second state is eliminated if $v_1 \leq v_2$.

The outcome of Phase II will be a set of routes covering all nodes in G exactly once with total minimum cost.

4.2.2 Computational Experiments:

The exact method was coded in Matlab developed by Math Works, Inc. The experimental tests consist of a Euclidean plane in which customer coordinates are uniformly distributed between 0 and 7, and travel times and costs equal distances. The 'time to start service' at each node is generated as a uniform random variable between 9:00 a.m. to 5:00 p.m. The time window at each node has two sides. Each side consists of a time window generated as a uniform random variable in the interval $[0,1]$. The service time at each node is assumed to be a uniform random variable generated between 30 minutes to 2 hours.

The coordinates of the depot were generated randomly between 3 and 4 using uniform distribution function. The time window at depot is set between 6:00 a.m. till 20:00 p.m. and the service time at depot is assumed to be zero.

The experimental results for graph G with different number of nodes (customers) are presented in Table 1. The experiments were tested on an Intel Pentium III, 600 MHZ. In Table 1, each set of customers (row) is built upon the previous row. For instance, for number of customers (nodes) equal to 10, we use the same randomly generated customers for $N=7$ and added 3 newly generated customers. For the first three sets of tests, where the problem size was relatively small (less than 15 nodes), the exact method was able to find the optimal solution.

Table 1. The exact method: computational experiments.

No of nodes	Optimum Cost	CPU time ⁶
7	13.68	0.44
10	22.47	2.31
15	38.04	156.65
20	NA ^a	NA

a) NA: The result couldn't be obtained

4.3 Genetic Algorithms for m-TSPTW

The exact method was applied on few sets of problems as shown in Table 1. Since the problem is NP-hard, the computational time for large size problems is fairly high. What we observed in our computational experiments is the DP algorithm for set covering

⁶ CPU time: is the time in seconds that has been used by the simulation program to obtain the final result.

(phase II) is computationally slow with problems involving around or more than 20 nodes.

Although the proposed method is capable of finding the exact solution for small size problems, the algorithm becomes computationally very costly for problems of medium or larger size. Hence, it is desirable to find a mechanism that results in a compromise between the quality of the solution and the computational time needed to obtain that solution. Meta-heuristic methods such as Tabu Search, Simulated Annealing (SA), and Genetic Algorithms (GA) offer such a mechanism that forces the algorithm out of the locally optimal solutions in their search for the globally optimal solution. These methods have been recently considered and applied to Set Covering Problems (SCP) with promising results, for instance see Beasley and Chu (1996), Lorena and Lopes (1997), Sen (1993), and Jacobs and Brusco (1994).

Since it is computationally prohibitive to find the optimal solution among the feasible set of solutions, we propose using a genetic algorithm for solving the second phase in order to find a ‘good’ solution for large size problems.

4.3.1 Methodology:

First, the sets of the routes found in Phase I are encoded in the form of matrix β presented in (9). Matrix β is an $n \times m$ matrix, where n is the number of customers in graph G and m in the number of the routes generated in phase I. Each element β_{ij} of this matrix has a binary value, i.e., it is either one or zero. The β_{ij} is one if route j visits node $i \in N$, and zero otherwise. Without loss of generality, we assume that the columns of β are ordered in decreasing order of number of ones in each column (nodes visited by the route j), and columns with equal number of ones are ordered in increasing order of cost. Equation (10) illustrates a typical matrix β for a graph of 4 nodes where 11 feasible routes were generated in Phase I.

$$\beta = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

A solution is defined as a set of routes (columns in β) that visited all the nodes (rows in β) in graph G exactly once. We used an m -bit binary string, which will be called chromosome thereafter, to represent a solution structure. A value 1 for the i th bit in the string implies that column i is in the solution set. Equation (11) shows two typical solutions (chromosomes) for matrix β in (10).

$$\begin{aligned} chrom1 &= [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0] \\ chrom2 &= [0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] \end{aligned} \quad (11)$$

To each chromosome a cost (*fitness function*) is associated which determines how 'good' each chromosome is. With binary representation, the fitness function ψ_j of chromosome j can be simply calculated by

$$\psi_j = \sum_{i=1}^m F_i \cdot \rho_{ij} \quad (12)$$

where ρ_{ij} is the value of the i th bit in the chromosome j , and F_i is the cost of traveling on route i calculated in phase I.

The initial population of chromosomes is generated by randomly assigning feasible route i to chromosome j . The route i is weighted more if it covers more of the remaining nodes with less total costs. Among all population of chromosomes two are "selected" for generating new chromosome (*offspring*). We used the binary tournament selection method by forming two groups of chromosomes with equal number of chromosomes in each group. The chromosomes are randomly placed in each group. One chromosome from each group with the best fitness function is selected to produce an offspring. *Crossover* and *mutation* operators are then applied on two selected chromosomes to generate the offspring. Let's assume that chromosomes j and k have been selected for producing the offspring l . By applying the crossover operator, the value of ρ_{il} (the i th bit in chromosome l) is set equal to ρ_{ij} by probability p , which is equal to

$$p = \frac{\psi_j}{\psi_j + \psi_k}, \quad (13)$$

and to ρ_{ik} with probability $(1-p)$. Obviously, if the values of ρ_{ij} and ρ_{ik} are equal, the value of ρ_{il} will be equal to this value, i.e., $\rho_{il} = \rho_{ij} = \rho_{ik}$. By applying the mutation operator, the value of ρ_{il} will be inverted by some small probability q . However, the new generated offspring, i.e. l th chromosome, may not be feasible. A heuristic method can be used to make the chromosome feasible (e.g., see Beasley and Chu 1996).

The newly generated offspring substitutes one of the chromosomes in the initial population with fitness function worse than the average fitness of all population. This procedure will continue till the termination criterion is met, which is a predetermined number of iterations.

The GA algorithm is summarized as follows:

- Step 1: Form matrix β ;
Generate the initial population.
- Step 2: Select two chromosomes for generating offspring.
- Step 3: Apply the crossover operator.
- Step 4: Apply the mutation operator.
- Step 5: Make the offspring feasible.
- Step 6: Calculate the fitness function associated to the offspring.

Step 7: Substitute one of the chromosome in the initial population with offspring.

Step 8: Repeat the steps 2-8 until the termination criterion is met. The best solution is the one with the minimum fitness function in the population.

It is worth mentioning that it is generally believed that GA is slow and would take time to find a high quality solution (Zalzala and Fleming P.J. 1997). The amount of computational effort required by this algorithm depends on the size of the problem, i.e., number of the nodes in graph G (number of the rows in matrix β) and the number of the generated feasible solutions in Phase I (number of the columns in matrix β).

4.3.2 Computational Experiments:

The hybrid GA method was also coded in Matlab. Table 2 shows the results of using the hybrid GA method for finding the best solution among all feasible solutions (the top number in each cell is the value of the objective function, whereas the lower number is the computational time in seconds, as supplied by Matlab). The same data sets generated in Subsection 4.2.2 were used here to evaluate the efficiency of the GA method. For each set of nodes, the genetic algorithm was applied 10 times in order to assess the reliability and repeatability of the algorithm. A comparison of Tables 1 and 2 indicates that the hybrid GA algorithm was able to find the optimum solution for 7, 10 and 15 number of nodes at every trial in a relatively short amount of time.

These promising results encouraged us to extend the number of nodes in graph G up to 100. Thus, we used the same methodology described in Subsection 4.2.2 to generate new set of nodes (customers). That is each set of customers (row) is built upon the previous row. Table 2 also shows the result of using the GA algorithm for finding a good solution to the graphs having more than 15 nodes.

Table 2. The hybrid GA method: computational experiments.

No of nodes	Best solution in each of 10 trials using GA									
7	13.68 ^a / 0.93	13.68/ 1.59	13.68/ 1.16	13.68/ 0.55	13.68/ 0.55	13.68/ 0.66	13.68/ 0.66	13.68/ 1.26	13.68/ 1.32	13.68/ 0.99
10	22.47/ 1.65	22.47/ 2.91	22.47/ 3.46	22.47/ 1.21	22.47/ 2.42	22.47/ 1.15	22.47/ 4.23	22.47/ 5.49	22.47/ 3.24	22.47/ 8.73
15	38.04/ 4.18	38.04/ 10.2	38.04/ 8	38.04/ 9.28	38.04/ 6.21	38.04/ 5.05	38.04/ 10.9	38.04/ 13.0	38.04/ 9.44	38.04/ 7.75
20	55.2/ 20.7	54.3/ 29.3	53.6/ 23.4	55.2/ 29.4	54.0/ 13.8	55.2/ 25.3	54.1/ 22.2	54.3/ 28.9	54.2/ 18.5	54.1/ 27.2
30	89.1/ 82.9	90.0/ 47	89.0/ 71	88.1/ 87.5	87.9/ 66.4	88.5/ 83.4	88.9/ 88.5	88/ 85.4	87.9/ 53	87.8/ 87.6
50	158/ 145	153/ 192	153/ 181	151/ 187	154/ 187	150/ 187	151/ 183	150/ 169	153/ 190	151/ 191
100	325/ 2457	325/ 2051	329/ 2066	329/ 2327	325/ 2304	324/ 2178	330/ 2122	335/ 2529	325/ 2343	324/ 2199

a) The best value obtained from GA method / CPU time.

4.4 Insertion method

Due to the fact that the m-TSPTW is an NP-hard problem, together with the reality that the practical problems might have hundreds, or even thousands, of customers with wide time windows, there is not much hope for finding an optimal algorithm that works in practice. Recall that even finding a feasible solution to the m-TSPTW problem is NP-hard (Savelsbergh 1985). Therefore, the development of heuristic algorithms for this class of problems is of primary interest (Solomon 1987). This is the main reason why the majority of research in this area (and related ones, i.e. VRP, VRPTW, DARP) has been focused on finding an algorithm that will find ‘good solution’ in reasonably short amount of time, e.g., see Solomon (1987), Jaw *et al.* (1986), Madsen *et al.* (1995), Lee (1992), and Cavo (2000).

In his paper, Solomon (1987) described a variety of heuristics. He conducted an extensive computational study of evaluating the performance of each heuristic. He found that several heuristics performed well in different problem environments; however, in particular insertion-type heuristic consistently gave very good results. In this Subsection, we also develop an insertion heuristic for large size problems. The effectiveness and more importantly the quickness of developed algorithm, which will be discussed later, encourage us to use the same heuristic in dynamic situations.

4.4.1 Methodology:

The main classifications of heuristic methods are: 1) tour construction heuristics and 2) tour improvement heuristics. In tour construction heuristic, an algorithm builds (constructs) a set of feasible routes. In insertion heuristic a set of routes is built by inserting one request at a time into one route. Here, we are developing a sequential insertion procedure in which one route is generated at a time, and new route is initialized when the current routes are full. First, the algorithm searches for *feasible* insertions of customers into active routes, and then an *optimization* procedure is used to find the best feasible insertion location.

Feasible insertion procedure: Let $r=\{o, l, \dots, i, j, j+1, \dots, n, d\}$ be a feasible solution (route) starting from origin $\{o\}$, visiting nodes in route r only once, and ending at destination $\{d\}$. Figure 5 illustrates a typical route r .

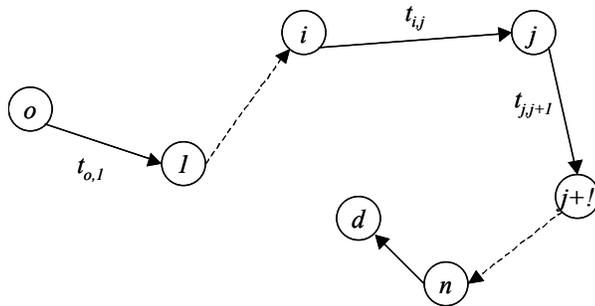


Figure 5: A graphical representation of a typical route.

Let A^r be the arc set associated with route (solution) r defined as $A^r = \{(i,j) \mid i,j \in r, \text{ and } j \text{ is visited immediately after } i\}$. Let c_{ij} and t_{ij} represent the cost and the time of traveling between two consecutive nodes i and j , respectively. Let also T_i be the time when service begins at node i . Recall that associated to each node $i \in r$ are a service time s_i and a time window $[a_i, b_i]$, where s_i is the duration of the time for a vehicle to be served at node i , and a_i and b_i are the earliest and the latest time to visit node i , respectively.

Assuming that node j is visited after node i , the *waiting time* at node j denoted by w_j is defined as the duration of time at node j the vehicle has to wait before being served, and is given by

$$w_j = \max(0, a_j - (T_i + s_i + t_{ij})). \quad (14)$$

Similarly, the *excess time* at node i denoted by Δ_i is defined as the portion of the time-window (at node i) between the latest time to visit node i , b_i , and the time that the service has started at node i , and is given by

$$\Delta_i = b_i - \max(T_i, a_i) \quad (15)$$

Figure 6 shows graphically the values of waiting time, excess time, service time, and time windows at each node of typical route r . Each arrow in Figure 6 indicates the traveling between pairs (i,j) in A^r .

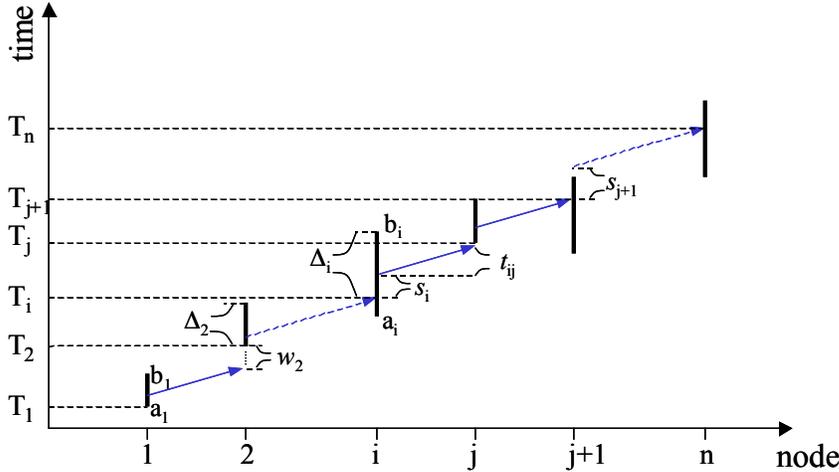


Figure 6: The graphical representation of waiting time, excess time, service time, and time windows at each node in a typical route.

Let δt_{ij} be the disturbance (changes from the nominal value) occurs in the traveling time between nodes i and j . The *feasibility margin* of the solution r to the changes in traveling time in link (i,j) is denoted by ϕ_{ij} and can be defined as the maximum value of δt_{ij} such that the solution r is still feasible. The *feasibility margin* of solution r with respect to the changes in traveling time of the link (i,j) can be computed as follows:

$$\varphi_{ij}^r = \underset{k \in \{j, j+1, \dots, n, d\}}{\text{Min}} \left(\Delta_k + \sum_{l=j}^k w_l \right). \quad (16)$$

The feasibility margin φ_{ij}^r implies how robust is the solution r to the changes on the traveling time between nodes i and j . The feasibility margin in (16) can also be calculated recursively as follows:

$$\begin{aligned} \varphi_{ij}^r &= w_j + \text{Min}(\Delta_j, \varphi_{j, j+1}^r) \quad \forall (i, j) \in A^r \\ \varphi_{d, d+1}^r &= M \end{aligned} \quad (17)$$

where M is a big number. Having defined the feasibility margin between each two nodes of route r , we shall now investigate the feasibility of inserting a node (customer) l between nodes i and j in route r . Node l can be inserted between each two consecutive nodes i and j if the following inequalities are satisfied:

$$t_{il} + w_l + s_l + t_{lj} - t_{ij} \leq \varphi_{ij}^r \quad (18)$$

$$T_i + s_i + t_{il} \leq b_l \quad (19)$$

Equation (18) indicates that the time changes in route i caused by inserting the node l between nodes i and j should be less than or equal to the feasibility margin between these two nodes. And, Equation (19) ensures that node l is visited before the latest time b_l .

Optimization procedure: Recall that to each route r a cost C^r is associated. The cost C^r is the summation of traveling costs between each two consecutive nodes in r . In other words, C^r is the total cost of traveling throughout all nodes of route r in the specified order. Let's assume that node l can be inserted between nodes i and j in route r . That is the inequalities (18) and (19) are met. The changes in cost C^r due to inserting node l between nodes i and j is denoted by ΔC_{ij}^r and is given by

$$\Delta C_{ij}^r = c_{il} + c_{lj} - c_{ij}. \quad (20)$$

To insert a node l in route r , we first find all feasible insertion locations according to (18) and (19). Among all feasible locations (arcs) in route r , a candidate arc $(i, j) \in A^r$ is one which results in minimum changes in cost C^r . In other words, we are interested in finding

$$\underset{(i, j) \in A^r}{\text{Min}} \Delta C_{ij}^r \quad (21)$$

We then examine all routes r to find the best locations among all feasible locations to insert node l . That is we are interested in finding the best candidate among all candidates with minimum cost of insertion, i.e.,

$$\underset{r}{\text{Min}} \left\{ \underset{(i, j) \in A^r}{\text{Min}} \Delta C_{ij}^r \right\} \quad (22)$$

Here, we summarize the insertion method:

Step 1: Initialization: construct the first route $R=1$.

Step 2: Select the next customer to be inserted

a) For all routes $r=1, \dots, R$

Find the best feasible location in route r for inserting the customer, according to (21)

b) If there is any feasible solution among all routes $r=1,..R$ find the route r^* results in minimum changes in the cost C^r

C) If there is no feasible location in Step 2-a for customer insertion, add another route to the set of routes, i.e. $R=R+1$, and insert the customers in the newly generated route.

Step 3: Termination: if there are no more customers to be inserted, Stop;

Otherwise go to Step 2.

4.4.2 Computational Experiments:

The same sets of generated customers in Table 2, for different number of customers in graph G , are used for evaluating the insertion method. Table 3 summarizes and compares the cost and the CPU time of the exact method, the hybrid Genetic Algorithm (GA) method, and insertion method. As it is shown in Table 3, exact method is efficient for relatively small size problems consisting of a few nodes. GA is capable of finding optimum solution for small size problems, and able to find 'good' solution for medium to large size problems (more than 30 nodes). It should be noted that in our computational experiments the maximum number of generated solutions (offspring) in GA was limited to 1000. Obviously, the larger the number of solutions is, the better the final result is. Finally, The heuristic insertion method was able to find relatively good solutions for medium to large size problems (more than 50 nodes); furthermore, the method is computationally very fast.

Table 3. Comparing exact, hybrid GA and insertion methods.

No of nodes	Dynamic Programming		Genetic Algorithm ^a		Insertion method	
	Cost	CPU time	Cost	CPU time	Cost	CPU time
7	13.68	0.44	13.68	0.97	17.69	0.11
10	22.47	2.31	22.47	3.45	28.23	0.11
15	38.04	156.65	38.04	8.38	45.64	0.22
20	NA ^b	NA	54.42	23.87	63.31	0.39
30	NA	NA	88.5	75.27	96.15	0.83
50	NA	NA	152.4	181.2	170.74	2.14
100	NA	NA	327.1	2257	330.91	8.62

a) In average (based on the results of 10 trials).

b) The result/data couldn't be obtained

4.5 Real data

The developed algorithms were tested using real examples containing a truck depot, container terminals, intermodal facilities and end customers supplied by the trucking company, Transport Express, in the City of Los Angeles. The city has some unique

specifications. It is adjacent to the Port of Los Angeles, the Port of Long Beach, and the Intermodal Container Transfer Facility (ICTF). Transport Express is a trucking company, which also does the warehousing and distributing. The loaded containers are brought to the depot, get unloaded, and then distributed among the end customers.

The real data were obtained through shadowing the dispatcher within a typical day. It is observed that, the dispatcher assigned the jobs to drivers (assignment problem), based on the availability of the drivers and the jobs. The priority in the assignment goes to those drivers who show up early at the depot. The dispatcher makes sure that the jobs are distributed evenly without concerns of optimizing a cost function. During the day, 64 containers were assigned to 22 drivers. This number includes 42 containers, which were moved between depot and the container terminals. This number shows the amount of similar activities at the depot.

Time windows for container delivery were very large, i.e., between 7:00 a.m. to 4:00 p.m. Only 10 jobs were associated with tighter time windows, i.e., before noon or after noon. This fact of having 65 nodes with wide time windows would actually render both the exact and the hybrid GA methods incapable of demonstrating any results. Thus, the only choice left was to use the insertion method. However, we want to re-emphasize the fact that by developing the exact and/or the hybrid method we haven't had any intention of optimizing the current ways of operations at the trucking companies or at the container terminals, but the future modes. We will discuss about the future modes later in this Subsection.

Table 4 shows the results obtained by applying the insertion method to the set of real data obtained from Transport Express. The cost in Table 4 is set equal to travel distance. In addition, the following assumptions were made before applying the insertion method algorithm.

The velocity of trucks: 30 miles/hour

Time to load or unload a container: 20 minutes

Table 4. Real Data solutions Comparison.

No. of nodes	Insertion method		Dispatcher	
	Cost	No. Vehicles Used	Cost	No. Vehicles Used
65	49.25	12	56.16	22

In our personal communication with people at Transport Express, we have learned that many container terminals in the Port of Los Angeles and Port of Long Beach have plans to enforce tight time windows (appointment system) at their gates in near future. The appointment systems are being considered as part of a solution for reducing terminal congestion and for the need to manage the future demand, i.e., the increasing number of containers at terminals and, consequently, the growing flow of trucks at gates. In the appointment system, a truck has to show up at the gates during a specific period of time to deliver or pick-up any specific container. It is worth mentioning that no standard has yet been established of how the appointment system should be implemented.

In the shed of this light, we made a scenario based on the real data we got from Transport Express. We assumed that the trucking company has to pickup and deliver the same cargo in a day with time windows assigned to each customer. We kept those time windows assigned to 10 jobs (described above) and we assign a time window with the length of 30 minutes to the other customers. We assumed that the velocity of trucks is 20 miles/hour and the loading or unloading time for any container is 30 minutes, and the cost is equal to travel distance. The results obtained by implementing the appointment system is shown in table 5 for the hybrid GA and for the insertion method.

Table 5. The result of applying appointment system on acquired Real Data .

No. of nodes	Insertion method		Hybrid GA method ^a	
	Cost	No. Vehicles Used	Cost	No. Vehicles Used
65	82.83	23	79.25	24.4

a) In average (based on the results of 10 trials).

4.6 Dynamic m-TSPTW

Recall that cargo movement is a dynamic problem; thus, any efficient algorithm should be quick, efficient and *dynamic*. In this subsection, we assume that the information of some or all of the customers is not available at the time of the route planning and scheduling. In other words, at the time of cargo assignment, carriers know only a portion of the customers to be served. The goal is to incorporate the new customer requests in existing pre-planned routes or to determine new routes to satisfy these requests.

In Section 3, we noted that, for a dynamic system, two methodologies could be employed: 1) insertion methods and 2) adaptation of static algorithms. The results of our computational experiments, presented above, indicate that only the insertion method is well suited to be considered as a dynamic algorithm for real time systems. The insertion method is very fast, compared to the other methods, and it is fairly efficient especially for the problems with large number of customers (see Table 3). Thus we are adopting the insertion method for dynamic systems to incorporate the new customers requests.

To measure the dynamism of a system, we adopt the concept of the Degree of Dynamisms introduced by Lund *et al.* (1996). The degree of dynamisms, denoted by DD , of a dynamic system can be defined as follows,

$$DD = \frac{N_D}{N_T} \times 100\%, \quad (23)$$

where N_D is the number of new customers and N_T is the total number of customers (advanced request customers as well as new customers). For advanced request customers, we use hybrid GA method for planning the routes for vehicles.

Figures 7, 8 and 9 show the efficiency of the insertion method in finding a good solution for a dynamic system with various degrees of dynamisms and different total number of customers.

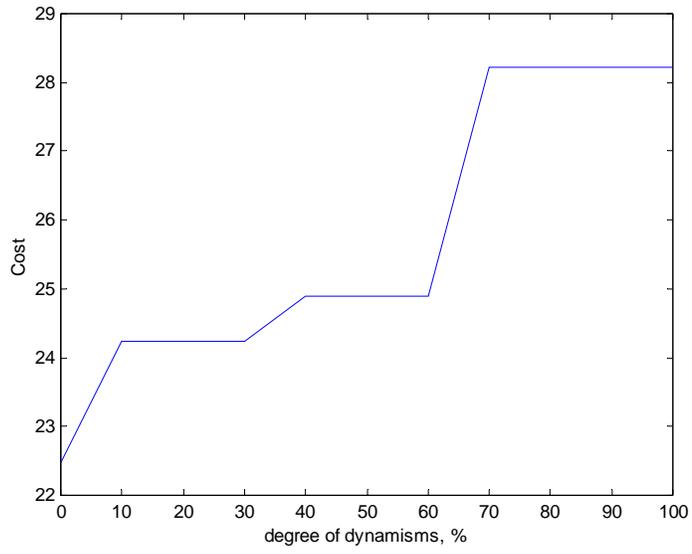


Figure 7: Route cost vs. the degree of dynamisms; $N_1=10$.

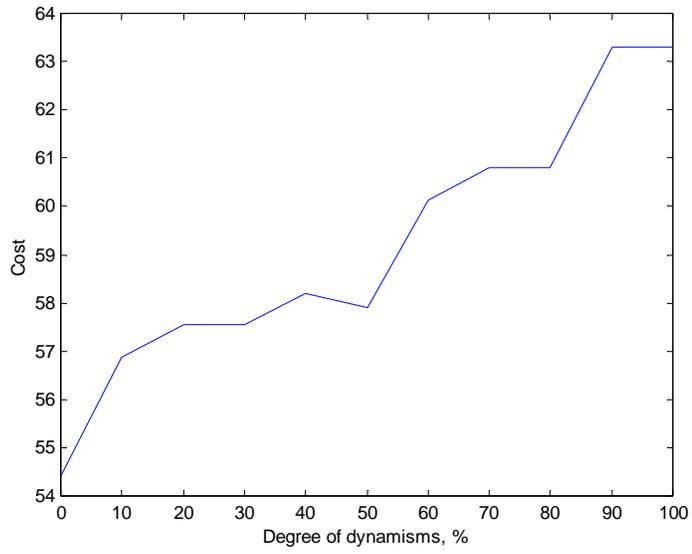


Figure 8: Route cost vs. the degree of dynamisms; $N_1=20$.

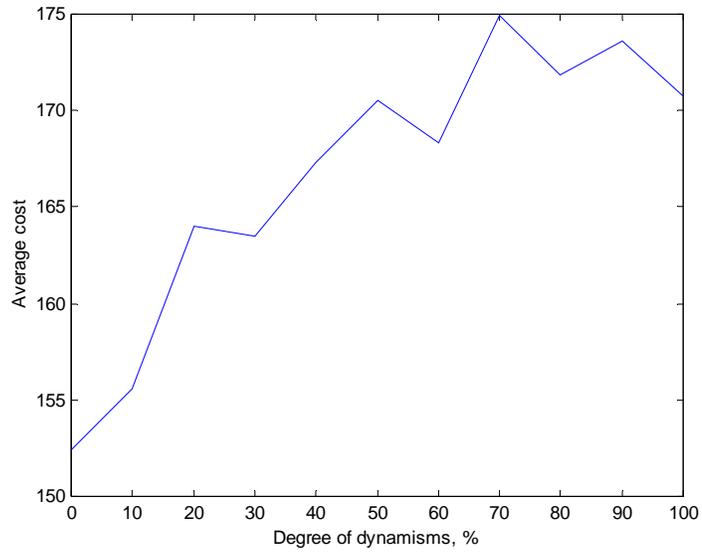


Figure 9: Route cost vs. the degree of dynamisms; $N_1=50$.

5 Conclusions and Recommendations

In this report, we investigated the cargo movement in metropolitan areas adjacent to ports. In particular, we were interested in improving the methods for trucks scheduling and route planning, where ISO containers need to be transferred between marine terminals, intermodal facilities, and end customers. The objective was to reduce empty miles, and to improve customer service. We showed that the container movements by trucks can be modeled as an asymmetric multi-Traveling Salesmen Problem with Time Windows (m-TSPTW). Moreover, we proposed three methodologies for solving the m-TSPTW:

- An exact two-phase Dynamic Programming (DP), for problems with less than 15 nodes and maximum window size 2 hours.
- A hybrid methodology consisting of DP in conjunction with genetic algorithms (GA), for problems with about 50 nodes and maximum window size 2 hours.
- A heuristic insertion method, for problems with more than 50 nodes with very wide window size.

We also investigated the dynamic problems in which the information of new customers becomes available after the set of routes has been determined. The results of our computational experiments indicate that the insertion method is well suited to be considered as a dynamic algorithm for real time systems. The insertion method is very fast, compared to the other methods, and it is fairly efficient especially for problems with large number of customers.

6 Implementation

The container movement operations are moving in a direction where, scheduling and routing will be coordinated in an effort to reduce empty travel times, reduce congestion at the terminals and reduce the overall cost. In this report we developed and tested several algorithms for scheduling and routing of trucks carrying containers for terminals and customers in a Metropolitan Area. These algorithms are developed to deal with problems faced in such environments and be computationally feasible especially in a dynamic environment. We interacted with Transport Express and use real data to evaluate the developed algorithms. The developed algorithms are ready for implementation by trucking companies. For this to be possible however, savings in cost and improved reliability in the working environment has to be established. This requires additional work and collaboration with a particular trucking company and is part of future work.

References

- Beasley J.E. and Chu P.C. (1996), "A genetic algorithm for the set covering problem," *European Journal of Operational Research*, vol. 94, pp. 392-404.
- Bellmore M. and Hong S. (1974), "Transformation of multi-salesmen problem to the standard salesman problem," *Journal of ACM*, vol.21, pp. 400-504.
- Bodin L., Golden B., Assad A., and Ball M. (1983), "Routing and scheduling of vehicles and crews: the state of the art," *Computers & Operations Research*, vol.10, no.2, pp. 63-211.
- Cavo R.W. (2000), "A new heuristic for the traveling salesman problem with time windows," *Transportation Science*, vol. 34, no. 1, pp. 113-124.
- Crainic T.G. and Laporte G. (1997), "Planning models for freight transportation," *European Journal of Operational Research*, vol. 97, pp. 409-438.
- Desrochers M., Lenstra J.K., Savelsbergh M.W.P., Soumis F. (1988), "Vehicle routing with time windows: optimization and approximation," In: Golden B.L., Assad A.A., (eds) *Vehicle Routing: Methods and Studies*, North Holland, Amsterdam, pp. 65-84.
- Desrochers M., Desrosiers J., and Solomon M.(1992), "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, pp. 342-354
- Desrosiers J., Dumas Y., Solomon M.M., and Soumis F. (1995), "Time constrained routing and scheduling," in: Ball M.O., Magnati T.L., Monma C.L., Nemhauser G.L., (eds.) *Network Routing, Handbooks in Operations Research and management Science, Volume 8*, INFORMS, Elsevier Science, pp. 35-130.
- Dumas Y., Desrosiers J., Gelinas E., and Solomon M.M. (1995), "An optimal algorithm for the traveling salesman problem with time windows," *Operations Research*, vol. 43, no. 2, pp. 367-371.
- Fisher M. (1994), "Optimal solution of vehicle routing problems minimum k-trees," *Operations research*, vol. 37, pp. 319-328.
- Fisher M. (1995), "Vehicle routing," in: Ball M.O., Magnati T.L., Monma C.L., Nemhauser G.L., (eds.) *Network Routing, Handbooks in Operations Research and management Science, Volume 8*, INFORMS, Elsevier Science, pp. 1-33.
- Garey M.R. and Johnson D.S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco.

- Gendreau M., Potvin J.Y. (1997), "Dynamic vehicle routing and dispatching", Technical report, CRT 97-38, Center de Recherche sur les Transport, universite de Montreal, November.
- Golden B.L. and Assad A.A. (1988), (eds.), *Vehicle Routing: Methods and Studies*, North Holland Publication, Amsterdam.
- Hall R.W. and Intihar, C. (1997) "Commercial Vehicle Operations: Government Interfaces and Intelligent Transportation Systems", California PATH Research Report UCB-ITS-PRR-97-12.
- Ioannou P.A., et al. (2000), Cargo Handling Technologies, Technical Report, Department of Electrical Engineering, University of Southern California.
- Jacobs L.W. and Brusco M.J. (1994), "A simulated annealing based heuristic for the set-covering problem," *Proceedings of Decision Sciences Institute*, vol. 2, pp. 1189-91.
- Jaw J.J., Odoni A. R., Psaraftis H. N., and Wilson N. H. M. (1986), "A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows," *Transportation Research*, vol. 20B, no. 3, pp. 243-257.
- Laporte G. (1992), "The vehicle routing problem: an overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, pp. 345-358.
- Lee M.S. (1992), New Algorithms for the m-TSPTW, Ph.D. Thesis, University of Maryland Collage Park.
- Lin S. and Kernighan B. (1973), "An effective heuristic algorithm for the traveling salesman problem," *Operations Research*, vol. 1, pp. 498-516.
- Lorena L. and Lopes L. S. (1997), "Genetic algorithms applied to computationally difficult problems," *Journal of Operational Research Society*, vol.48, pp. 440-445.
- Lund K., Madson O.B.G., and Solomon M.M. (1996), Vehicle routing problems with varying degree of dynamism, Technical Report, IMM, The Department of Mathematical Modelling, Technical University of Denmark.
- Kolen A.W.J., Rinnooy Kaan A.H.G., and Trienekens H.W.J.M. (1987), "Vehicle routing with time windows," *Operations Research*, vol. 35, no. 2, pp. 266-273.
- Madsen O.B.G., Ravn H.F., and Rygaard J.M. (1995), "A heuristic algorithm for a dial- aride problem with time windows, multiple capacities and multiple objectives," *Annals of operations Research*, vol. 60, pp. 193-208.
- Meyer, Mohaddes Associates, Inc. (1996), Gateway Cities Trucking Study. Gateway Cities Council of Governments Southeast Los Angeles County.

- Ng L., Wessels, R.L., Do D., Mannering F., and Barfield W. (1995) "Statistical Analysis of Commercial Driver and Dispatcher Requirements for Advanced Traveller Information Systems." *Transportation Research-C*, vol. 3, no.6, 353-369.
- Powell W.B., Jaillet P., and Odoni A. (1995), "Stochastic and dynamic networks and routing," in: Ball M.O., Magnati T.L., Monma C.L., Nemhauser G.L., (eds.) *Network Routing, Handbooks in Operations Research and management Science*, Volume 8, INFORMS, Elsevier Science, pp. 141-295.
- Powell W.B. (1996), "A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers," *Transportation Science*, vol. 30, no. 3, pp. 195-219.
- Psaraftis H.N. (1980), "A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem," *Transportation Science*, vol. 14, pp. 130-154.
- Psaraftis H.N. (1988) "Dynamic vehicle routing problems," in: Golden B.L., and Assad A.A. (1988), (eds.), *Vehicle Routing: Methods and Studies*, North Holland Publication, Amsterdam, pp. 223-248.
- Psaraftis H.N. (1995), "Dynamic vehicle routing: status and prospects", *Annals of Operations Research*, vol. 61, pp. 143-164.
- Reinelt G. (1994), *The traveling salesman problem: computational solutions for TSP applications*, Lecture Notes in Computer Science, Springer-Verlag.
- Ryan N.K.(1998), "The Future of Maritime Facility Designs and Operations," *Proceeding of 1998 Simulation Conference*, pp. 1223-1227.
- Savelsbergh M.W.P. (1985), "Local search in routing problems with time windows," *Annals of Operations Research*, vol. 4, no.1-4, pp. 285-305.
- Savelsbergh M.W.P. and Sol N. (1995), "The general pickup and delivery problem" *Transportation Science*, vol. 29, no. 1, pp. 17-29.
- Sen S. (1993), "Minimal cost set covering using probabilistic methods," *Proceeding of 1993 ACM/SIGAPP Symposium of Applied Computing*, pp. 157-164.
- 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.
- Taillard E. and Badeau P. (1997), "A tabu search heuristic for the vehicle routing problem with soft time windows," *Transportation Science*, vol. 31, no. 2.
- Taleb-Ibrahimi M., Castilho B. D., and Daganzo C. F. (1993), "Storage Capacity vs. Handling work in Container terminal," *Transportation Research B*, vol. 27B, no.1, pp. 13-32.

- Vickerman M. J. (1998), "Next-Generation Container Vessels: Impact on transportation Infrastructure and Operations," *TR News*, vol. 196, pp. 3-15, May-June.
- Wilson N.H.M., and Colvin N.H. (1977), "Computer dial-a-ride algorithms research project," Technical Report R-76-20, Department of Civil Engineering, Massachusetts Institute of Technology.
- Zalzala A.M.S., and Fleming P.J. (1997), *Genetic Algorithms in engineering Systems*, IEE control Engineering Series 55.