Algorithms for the Vehicle Routing Problem with Time Windows and the Location-Routing Problem

by

Suat Boğ

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Industrial Engineering

Koç University

July, 2006

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Suat Boğ

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Selçuk Savaş, Ph. D. (Advisor)

Metin Türkay, Ph. D. (Advisor)

Deniz Aksen, Ph. D.

Ceyda Oğuz, Ph. D.

Sibel Salman, Ph. D.

Date: _____

To my parents and brother

ABSTRACT

In this thesis we study two well-known routing problems. In the first, we solve the vehicle routing problem with time windows (VRPTW). The VRPTW involves designing a set of routes to serve all the customers without violating the capacity of vehicles and time windows constraints of the customers. We present a new branch-and-cut algorithm for solving the VRPTW. We apply a separation routine that detects all cycles in the nodes of the branch-and-cut tree in polynomial time and then tries to eliminate these cycles by checking the violated valid inequalities. A new branching strategy that branches on the cycles found in the node relaxations of the MILP model is also introduced. Computational experiments are performed on the Solomon test problems. So far the most successful exact algorithms for the VRPTW applied shortest path decomposition to the problem. When the proposed algorithm is compared with the other algorithms in the literature, it has been shown that decomposition based exact algorithms (i.e., column generation, Lagrangean relaxation) do not perform the best for the entire Solomon test problems, the performance of the proposed algorithm is better than decomposition based exact approaches on the long horizon test problems.

The second problem studied is the location-routing problem (LRP). The LRP consists of selecting a subset of locations among given candidate facility locations and determining the vehicle routes to visit all the customers from the selected facility locations. We propose a column generation algorithm for the solution of the capacitated location-routing problem in which we have multiple capacitated facilities and vehicles. The master problem in our column generation algorithm is a set-partition based formulation which proved to be useful for a variety of routing problems such as the vehicle routing problem with time windows. For the pricing subproblem we solve the elementary shortest path problem with resource constraints (ESPPRC). The resulting algorithm is tested on various LRP benchmark problems. The proposed column generation algorithm gives tight gaps for most of the benchmarks.

ÖZETÇE

Bu tezde literatürde yer alan iki rotalama problemi üzerinde çalışılmıştır. Bunlardan ilkinde, zaman çerçeveli araç rotalama problemini (ZÇARP) çözüyoruz. ZÇARP, araçların kapasitelerini ve müşterilerin belirlediği zaman çerçevesi kısıtlarını ihlal etmeden, müşterilere hizmetin sağlanacağı rotaların belirlenmesini içermektedir. ZÇARP'nin çözümü için yeni bir dal-kes algoritmasını sunuyoruz. Dal-kes ağacının düğümlerinde oluşan tüm döngülerin polinom zamanda belirlenmesi ve ihlal edilen geçerli eşitsizliklerin kontrolü ile yok edilmeleri için bir ayırma yöntemi uygulamaktayız. Ayrıca karışık tam sayılı doğrusal programın düğüm gevşetmelerinde bulunan döngüler üzerinde dallandırmayı sağlayan yeni bir dallandırma stratejisini de uygulamaktayız. Deneysel çalışmalar literatürde sık kullanılan Solomon'un test problemleri üzerinde yapılmıştır. Şu ana kadar ZÇARP'nin çözümü için en başarılı olan algoritmalar problemi basit problemlere ayrıştırıp, alt problem için en kısa yol problemini çözen algoritmalardır. Sunulan algoritma literatürdeki diğer algoritmalarla kıyaslandığında, ayrıştırmaya dayanan kesin çözümlü algoritmaların (örneğin, kolon üretimi, Lagrangean gevşetmesi) bütün Solomon test problemleri üzerinde en iyi performansı göstermedikleri tespit edilmiştir. Dal-kes algoritması geniş planlama horizonlu problemler üzerinde daha iyi bir performans göstermiştir.

Çalışılan problemlerin ikincisi yer belirleme-rotalama problemidir (YBRP). YBRP verilen aday tesis yerleri arasından en iyi yerlerin seçilmesini ve seçilen tesis yerlerinden tüm müşterilere hizmetin sağlanacağı araç rotalarının belirlenmesini kapsar. Çoklu ve kısıtlı kapasiteli tesis ve araçların bulunduğu yer belirleme-rotalama probleminin çözümü için bir kolon üretimi algoritmasını öneriyoruz. Kolon üretimi algoritmamızdaki ana problem birçok rotalama probleminde (örneğin, zaman çerçeveli araç rotalama problemleri) başarılı olduğu görülmüş küme bölme esasına dayalı bir formülasyondur. Fiyatlandırma alt problemi için kaynak kısıtlı basit en kısa yol problemini çözüyoruz (KKBEKYP). Sunulan algoritma literatürde verilen çeşitli YBRP kıyaslama problemleri üzerinde test edilmiştir. Önerilen kolon üretimi algoritması çoğu kıyaslama problemi için sıkı aralıklar bulmayı başarmıştır.

ACKNOWLEDGMENTS

First I would like to thank my supervisors Metin Türkay and Selçuk Savaş who have provided me with their guidance and suggestions. I also want to thank professors Ceyda Oğuz, Sibel Salman and Serpil Sayın for their valuable comments and support on my thesis.

I am grateful to members of my thesis committee for critical reading of this thesis and for their valuable comments.

I am also very thankful to my colleagues Eda, Zeynep, Canan, Hazal and Nesrin at Koç University for their loyal participation to the social events we had with them. Cem and Eren were not only my colleagues but also my office partners. They were always supportive and encouraging in any endeavor I undertook.

Finally I thank Ardi for providing me with support and encouragement during my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

VRPTW	Vehicle Routing Problem with Time Windows
TSP	Traveling Salesman Problem
LRP	Location-Routing Problem
MP	Master Problem (Integer)
RMP	Relaxed Master Problem
SP	Subproblem
ESPPRC	Elementary Shortest Path Problem with Resource Constraints
R	Set of feasible routes
LP	Linear Programming
MIP	Mixed Integer Programming

Chapter 1

INTRODUCTION

Today's competitive environment, global economy and demanding society require efficient and cost-effective flow of goods and information among the supply points and the demand points of every organization. Efficient design and operation of distribution system is an important subject that helps companies to stay competitive in the market and to improve customer service levels. According to Irish Business and Employers Confederation (IBEC) survey conducted in August 2004, the transportation and warehousing costs has the third highest priority for 309 responding companies in Ireland. Bodin et al. [6] reports that in 1980 approximately $400 billion were used in distribution cost in the United States and $15 billion in the United Kingdom. Transportation costs accounted for 15% of the U.S. GNP in 1978 [21]. The distribution costs can account for up to 70% of the value added costs of goods in the food and drink business [26]. Halse [28] reports that in 1989, 76.5% of all the transportation of goods was done by vehicles, which underlines the importance of routing and scheduling problems. All these examples highlight the need to increase efficiency of its logistics operations for considerable savings in a company. Therefore managers have to make crucial and challenging decisions regarding the number and location of facilities, management of vehicle fleet, routing and scheduling of vehicles, storage and delivery of raw materials, finished goods and information. Hence it is not surprising that solving different types of location, routing and scheduling problems has been an important and active research area in Operations Research.

In this thesis we contribute to these research efforts by designing algorithms for the solution of two well-known routing problems that appear in the design of logistics systems. The first problem is the Vehicle Routing Problem with Time Windows (VRPTW) and the second one is the Location-Routing Problem (LRP).

For the VRPTW we present a compact mixed integer linear programming (MILP) formulation with the objective function of minimizing the total distance traveled. We propose a branch-and-cut algorithm that aims to strengthen the lower bounds quickly by adding violated valid inequalities at the node relaxations of the tree. A new strategy that identifies and tries to eliminate all directed cycles which are generated between customers is applied. We also introduce a new branching strategy that is based on the directed cycles that are identified in the nodes of the tree.

For the LRP we first introduce a formal MILP model with multiple capacitated vehicles and facilities. The model reflects the interdependence between location and routing costs. We decompose the problem into a master problem and a subproblem. The master problem is a set partitioning problem with side constraints and the subproblem is an elementary shortest path problem with resource constraints (ESPPRC). We propose a column generation algorithm tailored to the solution of the LRP. This algorithm is the first column generation algorithm that is proposed to solve capacitated multiple facilities, multiple vehicles LRP in the literature. The algorithm tries to update the upper bound frequently and to keep the gaps between the upper and lower bounds in a manageable level. For the solution of the subproblem a new dynamic programming algorithm which solves the ESPPRC is proposed. The gaps are tightened since we solve an elementary shortest path problem hence eliminate all the k-cycles.

1.1 Outline of the Thesis

In Chapter 2 we present a relevant review of the existing literature for the VRPTW and LRP. In the following chapters we describe the problems in detail and present the solution procedures applied. In Chapter 3, we propose a branch-and-cut algorithm for the VRPTW and give the implementation details and results obtained on the Solomon benchmark problems. In Chapter 4, we propose a column generation algorithm for the LRP and give the implementation details and results obtained on the benchmark problems that are compiled by Barreto [3]. Finally, in Chapter 5 we give our concluding remarks.

Chapter 2

LITERATURE REVIEW

2.1 Vehicle Routing Problem with Time Windows (VRPTW)

VRPTW is an important logistics management problem that has applications in school bus routing, mail delivery, bank delivery, newspaper delivery, delivery of orders, and municipal waste collection [7, 52, 44, 25]. Efficient routing and scheduling of vehicles is shown to result in important savings for government agencies and industries in different sectors [68].

VRPTW is NP-Hard because it is a generalization of the Vehicle Routing Problem (VRP) which involves finding a set of routes that begins and ends at the same depot and satisfies all the customer demands. VRPTW has additional constraints that impose possible service start times for each customer. The time interval that specifies the possible starting times of service is called the time windows. This thesis considers the case with hard time windows where the vehicle must wait if it arrives to the customer location before the earliest possible start time. Furthermore, the latest start times cannot be violated.

There exists a host of methods for solving VRPTW. The methods can be classified into two major groups: exact methods and heuristic methods. Review of heuristic methods can be found in [16] and [67]. The exact methods can be grouped into two categories: decomposition techniques and cutting plane-based methods. Decomposition techniques can also be grouped under three major methods: column generation [14, 36, 49, 11, 8, 31], Lagrangian relaxation [39, 34] and variable splitting [28]. A common characteristic of the decomposition techniques is that the resulting subproblem is a shortest path problem with resource constraints. The VRPTW requires that each customer is visited only once, hence no cycles are permitted in the optimal solution. Therefore, the subproblem is the elementary shortest path problem with resource constraints (ESPPRC) and is NP-hard in the strong sense [36]. If the requirement for elementary paths is relaxed, there exists pseudo-polynomial algorithms for the subproblem [15]. In the nonelementary version of the subproblem, cycles are permitted, hence a customer can be visited more than once in a path [11, 49]. The

algorithm of Chabrier [8] applies an elementary shortest path decomposition. Irnich and Villeneuve [31] makes a compromise between solving elementary and nonelementary shortest path problem. They solve a nonelementary shortest path problem but forbid cycles of length smaller than or equal to k.

Prior to our work, there exists only two pure cutting plane approaches reported in the literature [33, 2]. Kallehauge and Boland [33] present a new formulation using only binary arc variables and they use a new class of strengthened path inequalities that eliminate time and capacity infeasible paths. They tested their algorithm on long horizon test problems. They solved the previously unsolved instance R208.50 in 53,815.2 secs using UltraSPARC III Cu 900Mhz with 384Gb RAM.

Bard et al. [2] use a two-index MILP formulation without applying any decomposition. In a branch-and-cut framework, they relax the original MILP formulation and try to reduce the gap by identifying the violated valid inequalities. However, their objective function is to minimize the number of vehicles; minimizing the total distance traveled is a secondary objective and is only estimated after the minimum number of vehicles are found.

In our formulation, the objective is to minimize the total distance traveled. Our observation is that while solving a series of LP relaxations in the nodes of the branch-and-bound tree, many cycles which should not exist in an optimal solution, form. Formation of cycles leads to weak lower bound values. The main goal of our algorithm is to detect cycles and try to eliminate them as quickly as possible. It is shown that elimination of cycles improves lower bounds quickly. Identifying the cycles in the nodes of the branch-and-bound tree and adding violated valid inequalities help to eliminate these cycles. Also we propose a new branching strategy that branches on the cycles found. If we cannot find a valid inequality that can eliminate a particular cycle in a node then we opt to branch on that cycle.

2.2 Location-Routing Problem (LRP)

Major decisions that affect the performance of a distribution system are the location of facilities, the allocation of customers to the facilities and the structure of the delivery system. These important decisions determine the cost of the distribution system and the level of customer service provided by the system. The main cost components incurred by the decisions of managers are the location costs and the transportation costs. Early models for construct-

ing distribution system inaccurately represented the transportation cost by considering that each delivery will be made by one vehicle to serve one customer on a straight-and-back basis. Hence, these models calculated the delivery costs by using the "moment sum" equation [60]. Moment sum equation can be valid under the assumption that full truck loads are carried from facilities to the customers. However, if customer demands are less than the capacity of the vehicle, routes that include several customers can be constructed. This is the case when the customers' demand are less than a truck load (LTL). LTL shipments are observed more frequently and vehicle routing problem should be incorporated to the location problem to have a more realistic model. The need to consider the vehicle routes while designing the distribution system reveals the interdependence between location problem and the vehicle routing problem. Early studies that identify the inaccurate representation of transportation costs and emphasize the importance of integrated location-routing problems (LRP) include [71, 19, 61, 62].

In the integrated location-routing models, facility location and vehicle routing decisions have to be given simultaneously. Therefore, LRP is a combination of two already difficult problems: Facility Location Problem (FLP) and Vehicle Routing Problem (VRP). Both FLP and VRP have been shown to be NP-hard [12, 35, 50], thus the LRP is NP-hard as well [69]. Research on LRP started in the late 1970s. Early work (prior to 1988) on LRP is summarized by Laporte [45]. A more recent survey can be found in [56]. Laporte [45] reviews the various mathematical models and exact solution algorithms proposed. Both two-index and three-index models have been proposed for the mathematical formulation of the LRP. Laporte and Nobert [46] used a two-index formulation to solve an uncapacitated single facility LRP in an exact approach with constraint relaxation and branch-and-bound algorithm. Using two-index formulation and exact approach, Laporte et al. [48, 47] extended their study for solving uncapacitated multi-facility LRP [48] and capacitated multi-facility LRP [47]. Instances up to 50 nodes could be solved for the uncapacitated instances. For the capacitated instances, the largest problem solved has 8 candidate facility locations and 20 customers. A three-index mixed integer formulation was proposed by Or and Pierskalla [58] for a practical size LRP (117 customers, 4 facilities) to locate regional blood banks to serve hospitals. Jakobsen and Madsen [32] and Madsen [51] solve a two level LRP for a newspaper delivery system. Perl [59], Perl and Daskin [60] studied the distribution system

of an international manufacturer and distributor in United States. They provided a three-index formulation and decomposed the problem into three subproblems and proposed a heuristic method that solves location and routing phases iteratively. Initial feasible solution was found by route-first location-allocation-second approach. Hansen et al. [29] modified the integer formulation of Perl and Daskin [60] and provided an improved model by using continuous flow variables. Hansen et al. [29] further provided a more effective heuristic method that resulted in better solutions for the benchmark problems when compared to [60]. Wu et al. [73] presented a similar heuristic method to solve the Location-Allocation Problem (LAP) and the Vehicle Routing Problem (VRP) iteratively. They used simulated annealing as the basis to search solutions for LAP and VRP subproblems. On some instances they achieved better solutions then [60] and [29]. Tuzun and Burke [69] used the route-first, location-allocation-second approach and applied tabu search metaheuristic in location and routing phases. They compared their results with [65] which used a savings algorithm in the routings phase and solves an LRP problem with uncapacitated facilities. Melechovský et al. [53] solve an LRP with non-linear facility costs. They use a variable neighborhood structure in the improvement phase. Tabu search metaheuristic is applied within the neighborhood movements.

It is observed that exact procedures are applied on two-index formulations. Three-index formulations are more complex, they define the feasible region with more variables and constraints and they have not been solved exactly in the literature. The number of studies that address exact solutions is less compared to the ones with heuristic approaches due to the complexity of the problem. Most of the heuristic approaches decompose the problem into subproblems and solve the subproblems iteratively by considering the interdependence between the subproblems. Computational comparisons with other papers are very limited since a variety of LRPs are solved in each paper and benchmark problems for general LRPs are not much and have not been compiled until recently.

In our study, we focus on a more general deterministic LRP that has multiple capacitated facilities, multiple homogeneous and capacitated vehicles. Our objective is to minimize routing cost and location cost. Location cost of the facilities includes fixed cost of establishment and variable cost of facility throughput. Routing costs are calculated by using the cost per mile parameter and the Euclidean distances. In the LRP literature, the only problem sets

that exactly match with the type of LRP we solve, are provided by Perl [59]. The three LRP benchmark problems of Perl are studied in [60], [29] and [73]. The second closest LRP data set is provided by Or [57] and is studied in [58]. In Or's LRP benchmark, facilities are uncapacitated. The other benchmark instances we focus are adapted from VRP literature [13, 23, 55, 64, 9] by Barreto [3] and further studied in [4]. Barreto et al. [4] uses clustering technique in a sequential heuristic algorithm. The algorithm starts by constructing group of customers considering the vehicle capacity limit. Then for each customer group (each cluster) the routing is determined optimally (by solving a TSP problem) if the group has 40 customers or less, otherwise a two-stage heuristic procedure is applied that uses a savings method and 3-opt local improvement procedure. At the last step, a single source capacitated location problem is solved by treating each route as a single customer. We propose an exact column generation algorithm for the solution of the benchmark problems. Column generation has been applied successfully for a variety of vehicle routing problems. However, we are aware of only one study on column generation for LRP that solves an uncapacitated LRP that has distance constraints on the routes [5]. Berger et al. [5] propose a new set of constraints that significantly improves the linear programming relaxation of the master problem. Our algorithm is the first column generation algorithm in the literature that is proposed to solve capacitated multiple facilities, multiple vehicles LRP.

Chapter 3

VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

The Vehicle Routing Problem (VRP) involves finding a set of routes that begins and ends at the same depot and satisfies all the customer demands. The VRPTW has additional constraints that impose possible service start times for each customer. The time interval that specifies the possible starting times of service is called the time windows. We study the case with hard time windows where the vehicle must wait if it arrives before the earliest possible start time. Furthermore, the latest start times cannot be violated. Our objective is to minimize the total distance traveled.

The organization of the chapter is as follows. In section 3.1, we give the problem description for the VRPTW. In section 3.2, we introduce our formal mixed integer linear programming formulation. In section 3.3, applied branch-and-cut algorithm is presented in detail. We describe the cycle detection routine, valid inequalities and our new branching strategy. In section 3.4, the implementation details and results are given.

3.1 Problem Description

The VRPTW is defined on a directed graph $G = (V, A)$, where A is the set of arcs and $V = \{0, 1, ..., N, N+1\}$ is the set of nodes. The customers are labelled from 1 through N and is represented by the set $C = \{1, 2, ...N\}$. The depot is shown by the nodes 0 and $N+1$ where node 0 is incident to only outgoing arcs and node $N+1$ is incident to only incoming arcs. The benefit of representing the depot twice is the elimination of cycles (occurring at the node relaxations) that include the depot. In this way we make sure that the cycles in the node relaxations will only exist within the customers. There is an associated arc travel time T_{ij} and travel cost c_{ij} for each arc $(i, j) \in A$. Since we minimize the distance, travel cost of each arc $(i, j) \in A$ is equal to the Euclidean distance between nodes i and j, that is $c_{ij} = d_{ij}$. Vehicle velocity (vel) is assumed to be 1, that is why, the travel time along the arcs is equal to the length of the arcs. Furthermore, it is assumed that triangular inequality

Cust No.	Xcoord.	Ycoord.	Demand (d)	Earliest Time (a)	Latest Time (b)	Service Time (f)
0	35	35	0	0	230	0
1	41	49	10	161	171	10
2	35	17	7	50	60	10
3	55	45	13	116	126	10
4	55	20	19	149	159	10
5	15	30	26	34	44	10
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
17	5	30	2	157	167	10
18	20	40	12	87	97	10
19	15	60	17	76	86	10
20	45	65	9	126	136	10
21	45	20	11	62	72	10
22	45	10	18	97	107	10
23	55	5	29	68	78	10
24	65	35	3	153	163	10
25	65	20	6	172	182	10
Vehicle Number (K)				Capacity (Q)		
25				200		

Table 3.1: Example Data for the VRPTW

holds for both travel time and travel cost, that is for all nodes $i, j, k \in V$, $T_{ij} \leq T_{ik} + T_{kj}$ and $c_{ij} \leq c_{ik} + c_{kj}$. With each customer there is an associated service time f_i and demand d_i. For the depot we assume that $d_0 = d_{N+1} = 0$. The service of customer i must start within the time window $[a_i, b_i]$ where a_i and b_i represent the earliest and the latest times that the service can begin. All vehicles leave node 0 within $[a_0, b_0]$ and $a_0 = b_0 = 0$. Similarly, all vehicles must return to node $N + 1$ within $[a_{N+1}, b_{N+1}]$ and $a_{N+1} = 0$ is assumed. The time interval that the vehicles must complete their service can be denoted by $[a_0, b_{N+1}]$ and referred to as the planning horizon. Each vehicle has a capacity Q and the sum of the demands of the customers at a given route cannot exceed the vehicle capacity. The upper bound on the number of vehicles located at the depot is taken as 25 as given by Solomon test problems. In order to understand the problem, an example data for the VRPTW is illustrated in the Figure 3.1. This data has the same structure with Solomon's VRPTW benchmarks.

3.2 Mathematical Model

VRPTW model is presented using the two-index variables x_{ij}. The variable x_{ij} is equal to 1 if a vehicle travels from node i to node j along arc $(i, j) \in A$, 0 otherwise. In the two-index model, x_{ij} $(\forall i, j \in V)$ variables are used instead of traditional three-index variables, x_{ijk} whose value equal to 1 if the vehicle k travels from node i to node j along arc $(i, j) \in A$, 0 otherwise. Based on our computational experience, variables x_{ij} are only defined if nodes i and j do not form an incompatible path $(i \nrightarrow j)$, and if the sum of their demand is less than vehicle capacity $(d_i + d_j \leq Q)$ and if $i \neq j$ (Incompatible paths are defined in section 3.3.3). Furthermore, we opted not to define the variables x_{i0}, $i \in V$ and x_{N+1j}, $j \in V$ which represents the arcs entering node 0 and the arcs leaving node $N + 1$ respectively. The U_i $(\forall i \in C)$ variables correspond to the load on the vehicle at departure from node i. Their values are expected to increase along a route since the model is designed as a pick-up problem, that is, each vehicle leaves the depot empty, collects items from customers and returns to the depot with a full load. However, with a slight change on the constraint set, the model can also be run for service type of delivery. The t_i $(\forall i \in C)$ variables determine the time that the service starts at customer i. Each customer is associated with an integer variable M_i. The M_i variables were first proposed by Miller-Tucker-Zemlin for the traveling salesman problem [54] and they control the sequence of customers on a route. The value of the M_i variable for the customer i gives the order of the customer i in the route. We formulate the MILP formulation using these four decision variable sets as follows:

Sets:

$V = \{0, 1, 2.., N, N + 1\}$, set of all nodes

$C = \{1, 2, .., N\}$, set of customers

Model Parameters:

N: Number of customers

K: Number of available vehicles

d_{ij}: Travel distance between nodes i and j; $i, j \, \epsilon \, V$

Q: Vehicle capacity

vel: Vehicle velocity

d_i: Demand of customer i; $i \, \epsilon \, C$

f_i: Service time for customer i; $i \, \epsilon \, C$

a_i: Earliest service start time for customer i; $i \,\epsilon\, C$

b_i: Latest service start time service for customer i; $i \,\epsilon\, C$

Decision Variables:

x_{ij}: 1 if the vehicle goes from node i to node j, 0 otherwise; $i \,\epsilon\, C \cup \{0\}, j \,\epsilon\, C \cup \{N+1\}$,

t_i: Service start times for each customer; $i \,\epsilon\, C$

U_i: Load at departure from customer i; $i \,\epsilon\, C$

M_i: Order of a customer i on the route; $i \,\epsilon\, C$

Minimize

$$\sum_{i \epsilon V} \sum_{j \epsilon V} d_{ij} x_{ij}, \tag{3.1}$$

subject to

$$\begin{aligned}
\sum_{j \epsilon C \cup \{N+1\}} x_{ij} &= 1, \qquad \forall i \,\epsilon\, C,\ i \neq j && (3.2)\\
\sum_{j \epsilon C \cup \{0\}} x_{ji} &= 1, \qquad \forall i \,\epsilon\, C,\ i \neq j && (3.3)\\
\sum_{i \epsilon C} x_{0i} &= \sum_{i \epsilon C} x_{iN+1}, && (3.4)\\
K_{LB} \leq \sum_{i \epsilon C} x_{0i} &\leq K_{UB}, && (3.5)\\
t_i + (T_{ij}) x_{ij} + (b_i - a_j)(x_{ij} - 1) &\leq t_j, \qquad \forall i, j \,\epsilon\, C, && (3.6)\\
M_i + N x_{ij} - M_j + U_i - U_j + Q x_{ij} &\leq Q - d_j + N - 1,\ \forall i, j \,\epsilon\, C, && (3.7)\\
d_i \leq U_i &\leq Q, \qquad \forall i \,\epsilon\, C, && (3.8)\\
a_i \leq t_i &\leq b_i \qquad \forall i \,\epsilon\, C, && (3.9)\\
x_{ij} &\in \{0,1\}, \qquad \forall i,\ j \in V \mid i \neq j && (3.10)
\end{aligned}$$

where $T_{ij} = f_i + d_{ij}/vel$.

The objective function is to minimize total distance traveled. Constraint (3.2) states that there is exactly one arc going out from each customer. Similarly, constraint (3.3) states that there is exactly one arc coming into each customer. Constraint (3.4) ensures that the number of vehicles leaving node 0 equals the number of vehicles returning to node $N + 1$.

In constraint (3.5), K_{LB} is the simplistic lower bound on the number of vehicles which is the $\lceil \sum_{i \in C} d_i/Q \rceil$ and K_{UB} is the upper bound on the number of vehicles. If K_{UB} is underestimated then optimal solution is cut off, on the other hand, an unnecessarily large K_{UB} may yield very loose relaxation gaps. We implemented a heuristic search procedure to set K_{UB}. The details of this procedure is given in Section 3.3. Constraint (3.6) coupled with constraint (3.9) defines the time windows restrictions for each customer. Note that if $x_{ij} = 1$ then it ensures that $t_i + f_i + d_{ij}/vel \leq t_j$ and if $x_{ij} = 0$ then the constraint reduces to $t_i - b_i \leq t_j - a_j$ where $t_i - b_i$ is always less than or equal to zero and $t_j - a_j$ is always greater than or equal to zero by constraint (3.9). In constraint (3.7), two valid constraints are combined; the first is $U_i - U_j + Qx_{ij} \leq Q - d_j$ that was proposed by Kulkarni and Bhave [42] and the second one is $M_i - M_j + Nx_{ij} \leq N - 1$, proposed by Miller-Tucker-Zemlin [54]. The subtour elimination constraints $U_i - U_j + Qx_{ij} \leq Q - d_j$ allow to check vehicle capacity constraints for each route simultaneously with subtours. $M_i - M_j + Nx_{ij} \leq N - 1$ constraints check only the subtours by controlling the sequence of the customers on a route. In fact using only one set of constraints is enough for defining the feasible region. Kulkarni and Bhave's constraints are the extended version of the Miller-Tucker-Zemlin constraints and performs better than using the Miller-Tucker-Zemlin constraints. The relaxations of both of these constraints do not provide strong lower bounds. Interestingly, our computational experience has shown that combining these two weak constraints and using the combined constraint decreases the run times 18% on the average.

3.3 Branch-and-Cut Algorithm

3.3.1 Algorithm Structure

The major problem is that we have very low LBs and therefore large gaps for the corresponding LP problems at the tree. Branch-and-cut algorithm helps to increase the lower bounds by adding the violated valid inequalities to the nodes of the tree. The outline of our branch-and-cut algorithm is given in Figure 3.1.

To calculate travel distance, (or equivalently travel time by assumption) we use the Euclidean distances between pairs of coordinates. By following the most commonly used method starting from Kohl [36], Kohl and Madsen [39], Kohl et al. [38], distances are calculated with one decimal point and truncation. We also represent d_{ij} values as integers,

so we multiply d_{ij} by 10. The final distance formula is given in equation (3.11).

$$d_{ij} = \lfloor 10\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor \tag{3.11}$$

In order to remedy the effect of increasing the length of each arc 10 times, we also modify time windows parameters a_i and b_i and service time f_i by multiplying them with 10. Triangular inequality assumption is guaranteed by adding 1 to all d_{ij} for $i \neq 0$. If we exclude the arcs that originate from node 0, there exists N arcs in an optimal solution. This fact is due to the flow balance equations. That is why, by adding 1 to all d_{ij} for $i \neq 0$ we increase the total cost by N, hence the optimal objective value (OBJ^*) is calculated as $(OBJ_{final} - N)/10$.

In the preprocessing step, time windows reduction techniques presented in Desrochers et al. [14] and Kontoravdis and Bard [40] are applied. Number of time windows reductions realized for each problem is given in Table 3.9 under the column named "TWR". Arc elimination is done if equation (3.20) is satisfied ($i \nrightarrow j$). Note that $N(N+1)$ arcs remain after eliminating the arcs that originate from node $N+1$ (x_{N+1j}, $j \in V$), the arcs that terminate at node 0 (x_{i0}, $i \in V$) and the arcs in which originating and terminating nodes are the same (x_{ii}, $i \in V$). If the number of arcs eliminated after preprocessing is E then the number of arcs that remains in our MILP formulation is given by $N(N+1) - E$ and this number is shown in Table 3.9 under the column named "Arcs".

Before processing the root node, we find an initial feasible solution to serve as the first integer solution and give this solution to CPLEX as MIP start values. CPLEX reevaluates this starting solution for integrality and feasibility. If it is integer-feasible, it will serve as a starting integer solution of our instance. The MIP start values are found by using Clarke and Wright Savings algorithm as a construction heuristic for our VRPTW and it is followed by a series of route improvement heuristics, namely 2-opt, 1-1 exchange, crossover and simulated annealing. Number of routes, equivalently the number of vehicles in our initial feasible solution forms the upper bound K_{UB} on the number of vehicles in constraint (3.5) of the original MILP formulation.

We add cuts using the "cutcallback" facility of CPLEX. The overhead of adding cuts at each node of the tree does not justify the benefits in terms of tree size and elapsed time. Therefore, we do not add the cuts at each node of the search tree, instead we follow a

strategy that searches for and applies cuts at every 100 nodes in the computer program.

3.3.2 Cycle Detection Routine

The lower bound generated after we relax the integrality constraints of the original MILP formulation is low. The main reason for this is the weak time window and subtour elimination constraints. Relaxation of the model results in many cycles being generated. When the x_{ij} values can take values between 0 and 1, they can easily violate the constraints and form cycles. We use depth first search (DFS) in order to identify all directed cycles in a node of the tree, that is we find all k node cycles, where $k \in \{2, .., N\}$. An outline of the procedure used is given in Figure 3.2. Among the instances that we solved, the total number of cycles found in the whole tree can be found in Table 1.

In DFS, edges are explored from the most recently discovered vertices, v, that still has unexplored edges. When all of v's edges have been explored, the algorithm backtracks to the vertex from which v was discovered. When DFS is used, four types of edges can be identified. *Tree edges* are the edges in the depth first forest. *Back edges* are the edges (u,v) connecting a vertex u to an ancestor v in a depth-first tree. *Forward edges* are non-tree edges (u,v) connecting a vertex u to a descendant v in a depth-first tree. *Cross edges* are all other edges. Given a digraph G and any DFS tree of G, tree edges, forward edges, and cross edges all go from a vertex of higher finish time to a vertex of lower finish time. Back edges go from a vertex of lower finish time to a vertex of higher finish time. Hence when a back edge is detected it means that a cycle is detected. When a back edge (u,v) is found it is necessary to backtrack the predecessor of node u until we return to node v to identify the nodes that form the cycle. There are two loops in the algorithm; the first one loops through all vertices and the second loops through all neighbors of a particular vertex. The time spent in the outer loop is $O(\mid V \mid)$. The loop through all neighbors of all vertices takes $O(\mid V \mid + \mid A \mid)$. Therefore the complexity of the procedure is $O(\mid V \mid + \mid A \mid)$. After identifying a cycle, we have the following information for that cycle: Nodes of the directed cycle, total demand of the cycle and total inflow and total outflow of the cycle.

Steps of Branch-and-Cut Procedure

Inputs: Coordinates, demand, service time and time windows of each node

Vehicle capacity and velocity

Output: Optimal solution of VRPTW

Step 1. Initialization

Get the inputs

Calculate distance matrix

Modify time windows and service time parameters

Step 2. Preprocessing

Time windows reductions

Arc elimination

Step 3. Find an initial feasible solution using heuristics

Step 4. Construct VRPTW model

Step 5. Start solving the branch-and-cut tree

At every 100 nodes:

- Identify incompatible paths and pairs using time windows of the customers (see Figure 3.3)

Add the violated incompatible pair and path inequalities to the model

- Use cycle detection routine to identify the directed cycles in the nodes (see Figure 3.2)
- Check if the cycles violate a valid inequality

Add the violated valid inequalities to the model

- Branch on a directed cycle (see Figure 3.4)

Step 6. Route construction using the optimum x_{ij} values

Figure 3.1: Branch-and-Cut Algorithm

Cycle Detection Procedure

Input: x_{ij} values obtained from the optimal solution of the relaxed MILP

Output: A list of directed cycles

Step 1. For all arcs x_{ij} that are greater than zero, set $x_{ij} = 1$.

Let $X^+ = \{(i,j) \in A \mid x_{ij} > 0\}$ be the set of arcs who are set to one.

Step 2. Generate adjacency list of the directed graph composed of the arc set X^+.

Step 3. *Outer loop of DFS:*

For all nodes u of the graph

color[u]=White,

Predecessor[u]=NULL,

For all nodes u of the graph

if(color[u]==white) then DFS(u)

Inner loop of DFS:

color(u) = Gray,

For all neighbors of the node u ($\forall\ v \mid adj(u) = v$)

if color[v] = Gray and Predecessor[u] $\neq v$ do

Return "(u,v) forms a Back Edge"

Trace back the predecessors of u till the node v and,

Output the k-cycle: $u \rightarrow v \rightarrow 1 \rightarrow 2 \rightarrow ...(k-2) \rightarrow u$

if color[v] = White

Predecessor[v] = u

DFS(v)

color[u] = Black;

Figure 3.2: Identification of Directed Cycles

Separation routines for subtour elimination inequalities, D_k^+ and D_k^- inequalities are based on finding the cycles created in nodes of the tree and checking for the violated valid inequalities for each cycle found.

3.3.3 Valid Inequalities

Valid inequalities are used in order to reduce the feasible solution space of the relaxed MILP formulation at the nodes of the tree. The valid inequalities we try to identify are based on the x_{ij} variables that represent the arcs of the directed graph $G = (V, A)$. Therefore, the general form of the valid inequalities that are used can be stated as:

$$\sum_{(i,j)\epsilon A} \alpha_{ij} x_{ij} \geq \gamma \tag{3.12}$$

A separation problem has to be solved in order to identify the violated valid inequalities. Solving a separation problem in our context consists of finding α_{ij} and γ values such that the inequality is a valid inequality and it is violated by the current solution x_{ij} at a node of the branch-and-cut tree, that is:

$$\sum_{(i,j)\epsilon A} \alpha_{ij} x_{ij} < \gamma \tag{3.13}$$

There are four types of inequalities considered in this study: subtour elimination inequalities, k-path inequalities, D_k^+ and D_k^- inequalities, incompatible path and incompatible pair inequalities.

Subtour Elimination Inequalities

Our formulation of the VRPTW prevents cycles that includes the depot. The cycles that exist within the customers is checked to identify whether they violate subtour elimination constraints. Subtour elimination constraints are based on the minimum vehicle requirements of each set whose cardinality are between 2 and N. For the Traveling Salesman Problem (TSP) they can be defined as in equations (3.14) and (3.15).

$$\sum_{i\epsilon S} \sum_{j\epsilon V/S} x_{ij} \geq 1 \qquad \forall S \subseteq C \tag{3.14}$$

$$\sum_{i \epsilon S} \sum_{j \epsilon S} x_{ij} \leq |S| - 1, \quad \forall S \subseteq C \tag{3.15}$$

These two equations are interrelated and can be used interchangeably. The first equation says that the outflow from any set of the customers should be at least one, that is, one vehicle should exist in each set (one vehicle requirement is strictly satisfied for the TSP context). The second comes from the fact that the sum of the total outflow from a set and the total inflow within the set must equal to the number of customers in the set. This requirement comes from the flow balance equations defined in our formulation which also exist in the TSP context. So if the outflow is greater than one, then the inflow within the set must be less than the number of nodes in the set minus one. We first check this basic one vehicle requirement for each directed cycle we discover in a node of the tree. If the basic subtour elimination constraint is not violated then we check the more sophisticated k-path inequalities as described in the next section.

k-Path Inequalities

For the VRP and VRPTW, the right hand sides of the equations (3.14) and (3.15) form a loose bound because of capacity and time windows (only for VRPTW) restrictions. Hence, the right hand side can be changed by $k(S)$ which denotes the minimum number of vehicles required to satisfy set S. The resulting valid inequalities are presented in equations (3.16) and (3.17). These inequalities are known as k-path inequalities in the literature because they require that at least k paths leave set S [38].

$$\sum_{i \epsilon S} \sum_{j \epsilon V/S} x_{ij} \geq k(S), \quad \forall S \subseteq C \tag{3.16}$$

$$\sum_{i \epsilon S} \sum_{j \epsilon S} x_{ij} \leq |S| - k(S), \quad \forall S \subseteq C \tag{3.17}$$

However, finding the minimum number of vehicles for a set is NP-Hard in the strong sense. Hence, Kohl et al. [38] and Larsen [49] focus on the separation of 2-path cuts. They first try to find the sets S for which outflow is less than 2, then for a given set S that satisfies $\sum_{i \epsilon S} \sum_{j \epsilon V/S} x_{ij} < 2$, they find $k(S)$ by solving a TSPTW feasibility problem. If $k(S)$ is greater than one, it means that the set requires at least two vehicles, that is, $k(S) \geq 2$.

Each cycle that we identify in a node of the tree form the candidate set S for a k-path inequality. Note that we first check the violated subtour inequalities, if we cannot find a violation, we then check the violation of a k-path inequality. For each set S (that is, for each cycle) we try to determine the minimum number of vehicles to service the customers in set S. In fact, this is equivalent to solving a VRPTW with the objective function $min \sum_{i=1}^{N} x_{0i}$ used in Bard et al. [2]. However, due to the computational experience we had during runs, we again use the objective of minimizing distance traveled and check the number of routes in the optimal solution for the set S. In summary, we solve a small VRPTW problem in the node of the tree and for this small problem we apply all different types of cuts we generate for the whole problem. The aim is to get the same performance. Also we apply this procedure to sets that have nodes between 8 and 15 which are good candidates to find a k-path inequality and which we can solve relatively fast. We give a time limit of 10 secs to solve each small VRPTW problem optimally.

D_k^+ *and* D_k^- *Inequalities*

D_k^+ and D_k^- inequalities (equations (3.18) and (3.19) respectively) are inherited from the asymmetric TSP literature and are proposed by Grotschel and Padberg [27]. The inequalities are based on lifting the basic subtour elimination constraints presented in equation (3.15). D_k^+ and D_k^- inequalities are proved to be facet defining for the asymmetric TSP polytope [20].

Let $\{i_1, i_2, ..., i_k\}$ be a directed cycle of customers with $k \geq 3$:

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} + x_{i_k i_1} + 2\sum_{j=2}^{k-1} x_{i_j i_1} + \sum_{j=3}^{k-1}\sum_{h=2}^{j-1} x_{i_j i_h} \leq k-1 \tag{3.18}$$

$$\sum_{j=1}^{k-1} x_{i_j i_{j+1}} + x_{i_k i_1} + 2\sum_{j=3}^{k} x_{i_1 i_j} + \sum_{j=4}^{k}\sum_{h=3}^{j-1} x_{i_j i_h} \leq k-1 \tag{3.19}$$

We find all directed cycles in the nodes of the branch-and-cut tree by using the procedure presented in Figure 3.2, and then check all the cycles of length greater than or equal to three for a violation of equation (3.18) and (3.19).

Incompatible Path and Incompatible Pair Inequalities

We incorporate the incompatible path and incompatible pair inequalities that were first suggested by Bard et al. [2]. For all customer nodes i and j, we check whether node i can precede node j or not by using the triangular inequality we assumed. If both equations (3.20) and (3.21) are satisfied, then it means that node i cannot precede node j ($i \nrightarrow j$) and node j can precede node i ($j \rightarrow i$). In such a situation, we call any directed path that goes from node i to node j as an incompatible path. Note that satisfaction of equation (3.20) means that node i cannot be an immediate predecessor of node j. However, since triangular inequality is satisfied between all customers, node i can never precede node j in any way as an immediate predecessor or not.

$$a_i + T_{ij} > b_j \quad \forall i, j \in C \tag{3.20}$$

$$a_j + T_{ji} \leq b_i \quad \forall i, j \in C \tag{3.21}$$

$$a_j + T_{ji} > b_i \quad \forall i, j \in C \tag{3.22}$$

If node i cannot precede node j ($i \nrightarrow j$) and also node j cannot precede node i ($j \nrightarrow i$) as given in equations (3.20) and (3.22), then node i and j form an incompatible pair ($i \nleftrightarrow j$) and therefore they cannot coexist in the same path.

We follow a similar approach presented in Bard et al. [2] for the separation of incompatible path and pair inequalities. However, we opt to find the violated incompatible path and pair inequalities in one combined routine outlined in Figure 3.3.

Consider a directed path $\overrightarrow{P} = \{i, k_1, k_2, .., k_{|P|-2}, j\}$ from node i to node j with $|P|$ nodes and $|P|-1$ arcs. If, for an incompatible path of nodes ($i \nrightarrow j$, $j \rightarrow i$), such a directed path $\overrightarrow{P}$ is detected in a node relaxation, then the following straightforward valid inequality can be used to restrict the total flow along the path:

$$x_{i,k_1} + x_{k_1,k_2} + ... + x_{|P|-2,j} \leq |P| - 2 \tag{3.23}$$

Similarly, for an incompatible pair of nodes ($i \nleftrightarrow j$), if a directed path from node i to node j or a directed path from node j to node i is detected in a node relaxation, then the following straightforward valid inequality can be used to restrict the total flow along the path in both directions:

$$x_{i,k_1} + x_{k_1,i} + ... + x_{|P|-2,j} + x_{j,|P|-2} \leq |P| - 2 \tag{3.24}$$

Incompatible Path and Pair Detection Procedure

Input: x_{ij} values obtained from the optimal solution of relaxed MILP in a node of the tree

Output: Violated incompatible path and pair inequalities

Step 1. For all customer nodes i and j, identify all (i,j)s that
satisfy $i \nrightarrow j$ property in the preprocessing step.

Step 2. For all arcs $(i, j) \in A$, set the weight of the arc as $1 - x_{ij}$
(during cut callback in CPLEX)

Step 3. Run the all-pairs shortest path algorithm of Floyd-Warshall, 1962.

Step 4. For all $i < j$

If $i \nrightarrow j$ and shortest path distance between node i and j is smaller than one

if $j \nrightarrow i$

add equation (3.24)

else add equation (3.23)

Else if $i \rightarrow j$, $j \nrightarrow i$

if shortest path distance between node j and i is smaller than one

add corresponding incompatible path inequality.

Figure 3.3: Identification of Incompatible Paths and Pairs

3.3.4 Branching Strategy

Several branching strategies have been proposed in the literature. Most commonly used strategies are branching on the number of vehicles (proposed first by [14]), branching on the flow variables x_{ij} and branching on time windows [24, 49, 2]. Our new strategy is based on the directed cycles we identify in the nodes of the tree. The fact that the total inflow within a set S in an optimal solution must be an integer between 0 and ($\mid S \mid -1$) is utilized. We select for branching the set for which its inflow is furthest from one of the closest integer

values. We specifically branch on the directed cycles that we could not eliminate by adding cuts or that do not violate any valid inequalities. In order to clarify the first situation, think of a directed cycle whose outflow is 0.8. In such a situation you naturally add a basic subtour elimination cut given in equation (3.14). However, this cut does not guarantee to break the cycle, you may have the same cycle in another node with outflow greater than or equal to one. In this case, we make our branching on this cycle as described in Figure 3.4. In the procedure given, the nodes of each cycle form a candidate set S to be selected for branching. $x(S) = \sum_{i \epsilon S} \sum_{j \epsilon S} x_{ij}$ represent the total inflow of a directed cycle found, $\underline{x(S)}$ is the largest integer that is smaller than $x(S)$ and $\overline{x(S)}$ is the smallest integer that is greater than $x(S)$.

Procedure for Branching on the Cycles

Input: List of directed cycle sets obtained from Cycle Detection Procedure (Figure 2) in a node of the tree

Output: Branching set

Step 1. Out of all the cycle sets found in a relaxation, select the ones that do not violate any valid inequalities or the ones that were discovered at a previous node.

Step 2. For each selected set S define $\triangle_S = max\{x(S) - \underline{x(S)}, \overline{x(S)} - x(S)\}$.

Step 3. Select set S^* as the branching set where $\triangle_{S^*}$ is maximum.

Step 4. Branch on set S^* as given in equations (3.25) and (3.26).

Figure 3.4: Branching on Directed Cycles

Equation (3.25) is the branching up direction and equation (3.26) is the branching down direction.

$$\overline{x(S^*)} \leq \sum_{i \epsilon S^*} \sum_{j \epsilon S^*} x_{ij} \leq |S^*| - 1 \tag{3.25}$$

$$0 \leq \sum_{i \epsilon S^*} \sum_{j \epsilon S^*} x_{ij} \leq \underline{x(S^*)} \tag{3.26}$$

For instance, the output of the directed cycles in a node relaxation could be as follows:

- Cycle No 1 is a 3-Cycle with total inflow 1.86 ,outflow 1.14 and total demand 51
- Cycle No 2 is a 10-Cycle with total inflow 7.25 ,outflow 1.38 and total demand 177
- Cycle No 3 is a 6-Cycle with total inflow 3.84 ,outflow 1.69 and total demand 75
- Cycle No 4 is a 6-Cycle with total inflow 3.84 ,outflow 1.69 and total demand 75
- Cycle No 5 is a 6-Cycle with total inflow 3.56 ,outflow 2.44 and total demand 73
- Cycle No 6 is a 3-Cycle with total inflow 1.05 ,outflow 1.37 and total demand 26

6 cycles are detected in this relaxation. For the 5^{th} set, the total inflow is furthest from one of the closest integer values ($\triangle_5 = max\{3.56-3,\ 4-3.56\} = 0.56$). The 5^{th} set is a cycle of length 6. Therefore, the branching up direction for the set is $4 \leq \sum_{i \epsilon S_5} \sum_{j \epsilon S_5} x_{ij} \leq 5$ and the branching down direction is $0 \leq \sum_{i \epsilon S_5} \sum_{j \epsilon S_5} x_{ij} \leq 3$.

3.4 Implementation Details

3.4.1 Test Problems

Solomon's benchmark problems [63] have been used for the computational study. With respect to customer locations, there are 3 sets of problems: the R-problems contain customers that are randomly located, in the C-problems the customers are located in clusters and in the RC-problems some customers are distributed randomly and some distributed in clusters. With respect to the planning horizon, the problems are divided into two groups: short planning horizon problems (87 instances) and long planning horizon problems (81 instances). R1, C1 and RC1 problem sets have short planning horizons and they allow approximately 5 to 10 customers at each route. R2, C2 and RC2 problem sets have long planning horizons and they may allow more than 30 customers in a route. These are less constrained problems. Therefore, they are more difficult to solve. Only recently, there have been attempts to solve them by using exact methods [33, 8, 34]. Original problem instances have 100 customers (56 instances), but instances with 25 customers (56 instances) and 50 customers (56 instances) have been also created by considering the first 25 and 50 customers.

3.4.2 Computational Platform

Computational experiments are performed on a HP xw9300 Workstation with an AMD Opteron Processor 2.59 GHz and 8.00 GB of RAM at the Systems Lab of Koç University, Istanbul. We report the total elapsed times for each instance.

We have used ILOG CPLEX version 9.1 [30]. The algorithm is implemented with Java API of ILOG CPLEX Concert Technology. Separation routines that are used to identify violated valid inequalities have been executed using CPLEX cut callbacks.

3.4.3 Computational Results

In order to assess the performance of the branch-and-cut procedure comprehensively, the algorithm is tried on both long and short horizon benchmark problems of Solomon (168 instances). We give the optimal results obtained for these instances in Table 3.9. In this detailed table, number of time windows reductions (TWR), the arcs remaining after the arc elimination process (Arcs), total number of nodes in the tree (Nodes), total number of cuts added (Cuts), number of set branchings made (SetBr), number of cycles detected (Cycles), number of vehicles in the optimal solution (Vehs), optimal objective value (OPT^*) and the run time of the branch-and-cut procedure (Time(s)) is given respectively. The default CPLEX optimality tolerance for MIPs is a gap of 0.01%. This means that CPLEX will exit either when the number of nodes remaining is zero or if the gap between the best node and the best integer solution is 0.01% or smaller. We have used the same optimality tolerance in our runs. Note that run times under the column Time(sec) are rounded to 2 decimal places. Hence times smaller than 0.005 secs are given as 0.00. The algorithm is terminated if the optimum solution could not be found in 12 hours.

The proposed branch-and-cut algorithm solved 41 problems within 1 seconds, 62 problems within 10 seconds and 94 problems within 30 minutes. We could solve 96 out of the 168 Solomon test problems. Out of 96 problems, 46 problems belong to the long horizon class and 50 problems belong to the short horizon class. Table 3.2 gives the summary of problems solved by the proposed branch-and-cut algorithm and average and median of solution times of the corresponding problems. Table 3.3 gives an overview of the results compared to leading algorithms in the literature.

In Table 3.5, 3.6 and 3.7 we compare average and median of the solution times of our

Table 3.2: The summary of results obtained by the branch-and-cut algorithm

	Solved Problems	**Number of Problems**	**Average Running Time**	**Median**
25 node	56	56	43.39	1.20
50 node	27	56	224.70	5.17
100 node	13	56	157.85	9.35
All Nodes	96	168	108.15	2.29

Table 3.3: The number of Solomon Problems Solved

	Solved Problems		
Author	**short horizon**	**long horizon**	**total**
Desrochers et al. [14]	50	not solved	50
Kohl et al. [38]	69	not solved	69
Chabrier [8]	63	41	104
Kallehauge and Boland [33]	not solved	45	45
Irnich and Villeneuve [31]	76	59	135
Kallehauge et al. [34]	73	46	119
Proposed Algorithm	50	46	96
Total Solved	83	63	146
Total Number of Problems	87	81	168

algorithm with [31], [33] and [34] for the long horizon problems. Note that it is difficult to compare the run times that are taken across different computational platforms. Different processor speeds and random access memories should also be considered. In Table 3.4 we give the different platforms used for the computational experiments by the researchers.

Compared to [31], [33], and [34], the conclusion for the long horizon problems is that our algorithm outperforms the other algorithms. For the 25 node and 100 node problems, the average and median of the solution times of the branch-and-cut algorithm is significantly better than the 3 algorithms. For the 50 node problems, the medians of solution times are still better, but the average of the solution times is high since we have one problematic instance compared to the others. This instance is C204.50 which we could solve in 1102.83 seconds. Irnich and Villeneuve [31], Kallehauge and Boland [33], and Kallehauge et al. [34] solved this instance in 1159.4, 214.02 and 402.2 seconds respectively. Note that one instance can make the average worse, but you may perform very well for the other instances. Hence, our computational experiments show that median of the solution times is a better indicator

Table 3.4: Platforms used for the computational experiments

	Machine	**CPU**	**RAM**
Kallehauge and Boland [33]	Sun Fire 15K	UltraSPARC III Cu 900 MHz	384 Gb
Kallehauge et al. [34]	Sun Fire 15K	UltraSPARC III Cu 900 MHz	384 Gb
Irnich and Villeneuve [31]	Dell Inspiron 7500	Intel Pentium III 600 MHz	512 Mb
Proposed	HP xw9300 Workstation	AMD Opteron Processor 2.59 GHz	8 Gb
Desrochers et al. [14]	Sun Spark 1 Workstation	not mentioned	not mentioned
Kohl et al. [38]	HP9000/829	100 MHz PA7200 processor	not mentioned
Chabrier [8]	not mentioned	Pentium IV 1.5 GHz	256 Mb

of solution efficiency compared to the average solution times.

Table 3.5: Minimum, maximum, average and median of solution time in seconds for the same 43 long horizon test problems solved by [31] and the proposed algorithm

		Irnich and Villeneuve [31]				Proposed Algorithm			
problem size	number of problems	min	max	average	median	min	max	average	median
25	25	0.7	3455.3	270.66	6.30	0.00	438.33	23.75	1.14
50	12	6.4	1159.4	229.31	112.50	0.02	1102.83	116.14	1.95
100	6	99.8	3620.4	1562.05	1441.60	0.11	1507.59	302.24	50.35
All	43	0.7	3620.4	439.31	36.90	0.00	1507.59	88.39	1.41

In Table 3.8, we compare average and median of the solution times of our algorithm with [14], [38], [31] and [34] for the short horizon problems. For the short horizon problems, it is observed that our algorithm is not competitive with the other leading algorithms. The median of the solution times of the proposed algorithm is better than [38] and [14], but it is worse than [31] and [34].

Note that Chabrier [8] do not report all the solution times instead only reports the solution times for new solutions found. But we should indicate that even for some instances (R204.25, R208.25, RC204.25, RC208.25) that is closed by [8] in 2005 and also for the instance R208.50 that is closed by [33] in 2005, the run times of our branch-and-cut algorithm are significantly better. For the instance RC208.25 that is closed by [8], the difference in running time seems impressive. The running time reported in [8] is 33785.3 seconds whereas the running time for the proposed algorithm is only 269.1 seconds. Also for the instance R208.50 the running time reported is 53815.2 seconds, however the running time of the

Table 3.6: Minimum, maximum, average and median of solution time in seconds for the same 40 long horizon test problems solved by [33] and the proposed algorithm

		Kallehauge and Boland [33]				Proposed Algorithm			
problem size	number of problems	min	max	average	median	min	max	average	median
25	24	0.15	1834.85	85.86	4.72	0.00	59.16	6.47	0.99
50	11	0.83	214.02	58.33	37.48	0.02	1102.83	109.79	1.33
100	5	11.48	569.74	346.12	418.99	0.11	197.02	61.16	38.76
All	40	0.15	1834.85	110.82	15.49	0.00	1102.83	41.72	1.29

Table 3.7: Minimum, maximum, average and median of solution time in seconds for the same 39 long horizon test problems solved by [34] and the proposed algorithm

		Kallehauge et al. [34]				Proposed Algorithm			
problem size	number of problems	min	max	average	median	min	max	average	median
25	23	0.2	515.3	57.17	5.70	0.00	438.33	24.28	1.14
50	10	0.8	402.2	70.23	14.35	0.02	1102.83	114.23	1.13
100	6	2	1205.9	302.24	50.35	0.11	1507.59	226.68	35.95
All	39	0.2	1205.9	86.60	7.20	0.00	1507.59	90.11	1.33

proposed algorithm is 31323.08 seconds. These newly closed instances by [8] and [33] in 2005 are given in bold in the Table 3.9.

Table 3.8: Minimum, maximum, average and median of solution time in seconds for the same 39 short horizon test problems

Author	**min**	**max**	**average**	**median**
Desrochers et al. [14]	5.8	1064.2	197.85	97.15
Kohl et al. [38]	0.23	30.71	4.81	1.70
Irnich and Villeneuve [31]	0.1	115.8	6.62	1.00
Kallehauge et al. [34]	0.1	8.4	0.77	0.25
Proposed Algorithm	0	1001.875	45.39	1.53

Table 3.9: Results on the Solomon's VRPTW Benchmark Instances

PID	TWR	Arcs	Nodes	Cuts	SetBr	Cycles	Vehs	OPT*	Time(sec)
C101.25	3	332	0	0	0	0	3	191.3	0.00
C102.25	8	473	798	43	0	15	3	190.3	2.00
C103.25	14	569	4917	213	2	106	3	190.3	11.69
C104.25	18	604	16864	286	13	363	3	186.9	45.27
C105.25	3	357	0	0	0	0	3	191.3	0.00
C106.25	3	336	0	0	0	0	3	191.3	0.00
C107.25	4	377	0	0	0	0	3	191.3	0.00
C108.25	5	431	50	0	0	0	3	191.3	0.44
C109.25	5	482	2938	20	5	59	3	191.3	8.47
C201.25	2	353	0	0	0	0	2	214.7	0.00
C202.25	8	498	125	1	0	2	2	214.7	0.39
C203.25	13	584	20	56	1	34	2	214.7	1.41
C204.25	17	622	230	47	0	49	2	214.5	2.69
C205.25	4	391	2	0	0	0	2	214.7	0.11
C206.25	3	415	0	0	0	0	2	214.7	0.11
C207.25	3	463	32	7	0	7	2	214.5	0.39
C208.25	4	438	211	2	0	5	2	214.5	0.83
R101.25	1	224	0	0	0	0	8	617.1	0.00
R102.25	7	404	8	0	0	0	7	547.1	0.53
R103.25	13	510	5949	104	3	162	5	454.6	16.28
R104.25	17	559	8443	188	2	240	4	416.9	30.52
R105.25	1	300	0	0	0	0	6	530.5	0.08
R106.25	7	457	39	0	0	0	5	465.4	1.06
R107.25	13	541	5613	146	1	187	4	424.3	21.39
R108.25	17	585	1483	65	0	78	4	397.3	6.69
R109.25	1	410	1	0	0	0	5	441.3	0.36
R110.25	3	526	378650	4270	400	10134	4	444.7	1001.88
R111.25	4	525	5959	95	7	145	4	428.8	12.83
R112.25	4	646	56873	129	309	2316	4	393.0	156.25
R201.25	2	397	0	0	0	0	4	463.3	0.05
R202.25	7	518	111	0	0	0	4	410.5	1.14
R203.25	13	596	4700	154	0	203	3	391.4	14.12
R204.25	**17**	**627**	**20862**	**540**	**1**	**744**	**2**	**355.0**	**59.16**
R205.25	2	488	167	6	1	6	3	393.0	0.63
R206.25	7	565	1807	91	0	51	3	374.4	4.28
R207.25	13	609	464	56	0	43	3	361.6	2.91
R208.25	**17**	**631**	**5789**	**374**	**0**	**485**	**1**	**328.2**	**19.83**
R209.25	3	545	1483	58	0	52	2	370.7	5.02
R210.25	1	560	1289	44	0	31	3	404.6	4.58

Continued on next page

Table 3.9 – continued from previous page

PID	TWR	Arcs	Nodes	Cuts	SetBr	Cycles	Vehs	OPT*	Time(sec)
R211.25	3	647	146890	677	211	4259	2	350.9	438.33
RC101.25	2	276	370	0	0	0	4	461.1	0.50
RC102.25	8	407	1253	13	0	35	3	351.8	2.64
RC103.25	13	518	69783	2472	117	8082	3	332.8	126.00
RC104.25	17	564	327	37	2	134	3	306.6	0.84
RC105.25	6	370	2450	15	1	41	4	411.3	2.84
RC106.25	3	387	7361	48	8	142	3	345.5	5.61
RC107.25	5	513	522	8	0	16	3	298.3	0.89
RC108.25	9	616	535	10	0	18	3	294.5	0.98
RC201.25	3	401	2	0	0	0	3	360.2	0.12
RC202.25	8	522	265	1	0	2	3	338.0	1.25
RC203.25	13	596	18887	248	12	522	3	326.9	34.53
RC204.25	**17**	**627**	**365**	**7**	**0**	**13**	**3**	**299.7**	**0.88**
RC205.25	4	483	44	0	0	0	3	338.0	0.53
RC206.25	3	492	87	0	0	0	3	324.0	0.56
RC207.25	5	561	102	0	0	0	3	298.3	0.77
RC208.25	**6**	**650**	**133158**	**86**	**209**	**7490**	**2**	**269.1**	**379.28**
C101.50	5	1216	0	0	0	0	5	362.4	0.20
C102.50	16	1730	3132	404	1	96	5	361.4	16.78
C103.50	26	2163	77330	8704	51	8257	5	355.8	932.59
C104.50	26	2163	77330	8704	51	8257	5	355.8	1028.92
C105.50	5	1333	0	0	0	0	5	361.9	0.08
C106.50	6	1272	0	0	0	0	5	362.4	0.11
C107.50	8	1443	0	0	0	0	5	361.9	0.09
C108.50	9	1642	2011	3	4	39	5	361.9	5.17
C109.50	9	1895	140747	652	364	8076	5	361.6	1623.45
C201.50	2	1339	0	0	0	0	3	360.2	0.02
C202.50	14	1890	128	0	0	0	3	360.2	1.33
C203.50	25	2253	1888	255	0	70	3	359.8	30.20
C204.50	40	2505	57824	1137	48	2632	2	350.1	1102.83
C205.50	4	1476	0	0	0	0	3	359.8	0.38
C206.50	3	1589	8	0	0	0	3	359.8	0.74
C207.50	4	1814	306	1	1	13	3	359.6	2.61
C208.50	5	1691	258	5	0	12	2	350.5	2.58
R101.50	2	812	0	0	0	0	13	1045.3	0.06
R102.50	14	1435	19004	743	16	841	12	916.0	233.89
R105.50	4	1066	962	5	0	0	9	899.3	5.98
R201.50	5	1518	0	0	0	0	6	791.9	0.73
R208.50	**40**	**2525**	**489594**	**14867**	**588**	**50902**	**2**	**487.7**	**31323.08**
RC101.50	5	832	188293	1530	1779	4809	8	944.0	463.70

Continued on next page

Table 3.9 – continued from previous page

PID	TWR	Arcs	Nodes	Cuts	SetBr	Cycles	Vehs	OPT*	Time(sec)
RC104.50	40	2232	6066	9426	89	498	5	529.5	137.44
RC201.50	5	1495	137	0	0	1	5	686.7	0.92
RC202.50	17	1971	22844	539	3	1198	5	625.8	186
RC205.50	8	1834	9373	161	6	432	5	644.6	65.34
C101.100	9	4512	0	0	0	0	10	827.3	0.33
C102.100	31	6578	438441	64996	4	17855	10	820.3	35683.89
C105.100	9	5050	0	0	0	0	10	821.2	0.33
C106.100	12	5421	2089	37	13	140	10	827.3	10.70
C107.100	14	5617	0	0	0	0	10	818.9	0.28
C108.100	16	6342	1894	8	2	22	10	818.9	68.28
C201.100	3	5221	0	0	0	0	3	589.1	0.11
C202.100	28	7350	0	0	0	0	3	589.1	8.00
C206.100	7	6212	4323	112	3	332	3	586.0	61.94
C207.100	7	6578	1204	32	0	65	3	585.8	38.76
C208.100	10	6665	4055	216	0	603	3	585.8	197.02
R101.100	4	3241	13	0	0	0	20	1637.7	0.88
RC201.100	7	5918	29118	900	2	2909	9	1261.8	1507.59

Chapter 4

LOCATION-ROUTING PROBLEM

The location of the facilities in the vehicle routing problem has a great effect in the total cost of the distribution system. Similarly, consideration of the possible vehicle routes while solving a facility location problem has significance in minimizing the total system cost. Therefore, it is necessary to consider vehicle routing and facility locations simultaneously. However, it is argued that the horizon of these two problems differ. The routing problems are solved for short time horizons (even daily) and they are on the tactical and operational level, but the location problems are solved for long time horizons and they are on the strategic level. In spite of this, there are many studies that give examples of practical applications that combine location and routing decisions. Blood bank location [58], newspaper distribution [32], post box location [43], waste collection [41], goods distribution [60] and parcel delivery [70] are just some of the examples. The problem that integrates the facility location and vehicle routing problems together is the combined location-routing problem (LRP) [71, 59]. The LRP consists of selecting a subset of locations among given candidate facility locations and determining the vehicle routes to visit all the customers from the selected facility locations. The objective in LRP is to minimize the sum of the fixed costs, variable costs and delivery costs such that capacities of selected facility locations and vehicles are not exceeded while the demand of each customer is satisfied. The fixed costs include the costs of establishing depot on a candidate location and the variable costs depend on the per unit throughput to a customer from the selected facility.

The LRP reflects the interdependence between location costs and routing costs [71, 62]. Hence, it is a combination of two already difficult problems: Facility Location Problem (FLP) and Vehicle Routing Problem (VRP). Both FLP and VRP have been shown to be NP-hard [12, 35, 50], thus the LRP is NP-hard as well.

The organization of the chapter is as follows. In section 4.1, we give the problem description and in section 4.2, we introduce the mixed integer linear programming formulation for

the LRP. Proposed column-generation algorithm is presented in detail in section 4.3. We discuss the master problem and the pricing subproblem. We present the dynamic programming algorithm developed for the pricing subproblem. At the end of this section, we give the overall algorithm and explain the steps of the algorithm. In section 4.4, the implementation details and results are given.

4.1 Problem Description

The LRP model developed in this thesis uses some of the modeling concepts of VRPTW. It is shown in Chapter 3 of this thesis that the VRPTW model is very efficient in finding optimal solutions. Since VRPTW model is not directly applicable to LRP, some modifications are made on this model. The LRP is considered on a directed graph $G = (V, A)$, where A is the set of arcs and $V = \{1, ..., N, N+1, .., N+M, N+M+1, .., N+2M\}$ is the set of all nodes. The set of customers are labelled from 1 through N, $C = \{1, 2, ...N\}$ and the set of candidate locations are labelled from 1 through M, $J = \{1, 2, ...M\}$. The single depot in the VRPTW was represented twice, node 0 was the source node and node N+1 was the sink node. We had only outgoing arcs leaving node 0 and incoming arcs entering node N+1. A similar approach has been applied for the LRP. The M candidate facility locations are represented twice. Therefore, we form two sets of locations, one is the set of departure locations, $D = \{N+1, .., N+M\}$ and second is the set of arrival locations, $A = \{N+M+1, .., N+2M\}$. As an example, the corresponding arrival location for the departure location N+1 becomes N+M+1 in this representation. The set D is incident to only outgoing arcs and the set A is incident to only incoming arcs. Each candidate facility location j has capacity T_j. Total demand of customers assigned to a facility j cannot exceed the facility capacity T_j. There is a fixed cost, FC_j, for establishing a depot on each candidate facility location j and a variable cost, VC_j, that depends on the per unit throughput from each candidate location j. Another cost component is the arc travel cost c_{ij} for each arc $(i, j) \in A$. Travel cost of each arc $(i, j) \in A$ is calculated by multiplying cost per mile (CM) parameter by the Euclidean distance between nodes i and j ($c_{ij} = CM * d_{ij}$). There is an associated demand, d_i, with each customer. Each vehicle has a capacity Q and the sum of the demands of the customers at a given route should not exceed the vehicle capacity. The number of vehicles available (K_{UB}) is assigned as given if it is mentioned by

the corresponding benchmark case, otherwise it is taken as 25. In order to understand the problem better, an example data for the LRP is illustrated in the Figure 4.1.

Facility No.	Xcoord.	Ycoord.	Capacity (T)	Fixed Cost (FC)	Variable Cost (VC)
1	24	24	500	50	0.02
2	26	21	500	50	0.02
3	33	21	500	50	0.03
4	22	24	500	50	0.03
5	32	22	500	50	0.04
Vehicle Number (K)			25		
Vehicle Capacity (Q)			300		
Cost Per Mile (CM)			0.75		

Cust No.	Xcoord.	Ycoord.	Demand
1	23	35	125
2	45	43	84
3	12	45	60
...	...	...	...
...	...	...	...
...	...	...	...
34	30	16	160
35	10	40	40
36	34	60	80

Figure 4.1: Example Data for the LRP

4.2 Mathematical Model

The LRP model is presented using the 2-index variables x_{ij} as in the case of the VRPTW. The variable x_{ij} is equal to 1 if a vehicle travels from node i to node j along arc $(i, j) \in A$ and 0 otherwise. The variables x_{ij} are defined if sum of the demands of node i and j is less than vehicle capacity ($d_i + d_j \leq Q$) and if $i \neq j$. Furthermore, we do not define the variables x_{ij} if $i \in V$ and $j \in D$, because no arcs can enter to the set of departure locations. Similarly we do not define x_{ij} if $i \in A$ and $j \in V$, because no arcs can leave arrival locations. The y_j ($\forall j \in J$) variables are defined for each candidate facility location j. The variable y_j is equal to 1 if candidate facility j is selected; otherwise it is 0. The z_{ij} ($\forall i \in C, j \in J$) variable is equal to 1 if customer i is assigned to the facility j and 0 otherwise. The U_i ($\forall i \in C$) variables correspond to the load on the vehicle at departure from node i. Their values are

expected to increase along a route since the model is designed as a pick-up problem, that is, each vehicle leaves the depot empty, collects items from customers and returns to the depot with a full load. However, with a slight change on the constraint set, the model can also be modified for service type of delivery. The LRP is formulated as a MILP problem as follows:

Sets:

$J = \{1, 2, .., M\}$, set of candidate locations

$D = \{N + 1, N + 2, .., N + M\}$, set of departure locations

$A = \{N + M + 1, N + M + 2, .., N + 2M\}$, set of arrival locations

$C = \{1, 2, .., N\}$, set of customers

Model Parameters:

N: Number of customers

M: Number of candidate facility locations

K: Number of available vehicles

d_{ij}: Travel distance between nodes i and j; $i, j \,\epsilon\, V$

Q: Vehicle capacity

CM: Cost per mile

d_i: Demand of customer i; $i \,\epsilon\, C$

T_j: Capacity of candidate facility j; $j \,\epsilon\, J$

FC_j: Fixed cost of candidate facility j; $j \,\epsilon\, J$

VC_j: Variable cost of candidate facility j; $j \,\epsilon\, J$

Decision Variables:

x_{ij}: 1 if the vehicle goes from node i to node j, 0 otherwise; $i \,\epsilon\, C \cup D, j \,\epsilon\, C \cup A$,

y_j: 1 if the candidate facility j is selected, 0 otherwise; $j \,\epsilon\, J$

z_{ij}: 1 if customer i is assigned to facility j, 0 otherwise; $i \,\epsilon\, C, j \,\epsilon\, J$

U_i: Load at departure from customer i; $i \,\epsilon\, C$

Minimize

$$\sum_{j\epsilon J} FC_j y_j + \sum_{j\epsilon J}\sum_{i\epsilon C} VC_j d_i z_{ij} + \sum_{i\epsilon C\cup D}\sum_{j\epsilon C\cup A} c_{ij}x_{ij} \tag{4.1}$$

subject to

$$\begin{aligned}
\sum_{j\epsilon C\cup A} x_{ij} &= 1, \qquad \forall i\,\epsilon\, C, && (4.2)\\
\sum_{j\epsilon C\cup D} x_{ji} &= 1, \qquad \forall i\,\epsilon\, C, && (4.3)\\
\sum_{i\epsilon C} x_{ji} &= \sum_{i\epsilon C} x_{i(j+M)}, \quad \forall j\,\epsilon\, D, && (4.4)\\
K_{LB} \le \sum_{j\epsilon D}\sum_{i\epsilon C} x_{ji} &\le K_{UB}, && (4.5)\\
U_i - U_j + Qx_{ij} &\le Q - d_j, \qquad \forall i,j\,\epsilon\, C, && (4.6)\\
d_i \le U_i &\le Q, \qquad \forall i\,\epsilon\, C, && (4.7)\\
\sum_{i\epsilon C} d_i z_{ij} &\le T_j y_j, \qquad \forall j\,\epsilon\, J, && (4.8)\\
z_{ij} &\le y_j \qquad \forall i\,\epsilon\, C,\ \forall j\,\epsilon\, J, && (4.9)\\
\sum_{j\epsilon J} z_{ij} &= 1, \qquad \forall i\,\epsilon\, C, && (4.10)\\
x_{i(j+N+M)} &\le z_{ij}, \quad \forall i\,\epsilon\, C,\ \forall j\,\epsilon\, J, && (4.11)\\
x_{(j+N)i} &\le z_{ij}, \quad \forall i\,\epsilon\, C,\ \forall j\,\epsilon\, J, && (4.12)\\
x_{ij} + z_{ik} - z_{jk} &\le 1, \quad \forall i,j\,\epsilon\, C,\ \forall k\,\epsilon\, J, && (4.13)\\
x_{ij} + z_{jk} - z_{ik} &\le 1, \quad \forall i,j\,\epsilon\, C,\ \forall k\,\epsilon\, J, && (4.14)\\
x_{ij} &\in \{0,1\}, \qquad \forall i\,\epsilon\, C\cup D,\ j\,\epsilon\, C\cup A \mid i\neq j && (4.15)\\
z_{ik} &\in \{0,1\}, \qquad \forall i\,\epsilon\, C,\ \forall k\,\epsilon\, J && (4.16)\\
y_j &\in \{0,1\}, \qquad \forall j\,\epsilon\, J && (4.17)
\end{aligned}$$

The objective function is to minimize the total cost which is composed of the sum of fixed costs of establishing depots, the sum of variable costs that depend on each depot's throughput and the sum of delivery costs. Constraints (4.2) state that there is exactly one arc leaving each customer. Similarly, constraints (4.3) state that there is exactly one arc coming into each customer. Constraints (4.4) ensure that the number of vehicles leaving a departure depot equals to the number of vehicles returning to its corresponding arrival

depot. Constraint (4.5) enforces that the total number of vehicles used should not exceed the upper bound on the number of vehicles and it should be greater than K_{LB} which is calculated as $\lceil \sum_{i \in C} d_i/Q \rceil$. In constraint set (4.6), the valid inequality that was proposed by Kulkarni and Bhave [42] is used in order to eliminate cycles that can occur between customers. Constraints (4.7) give the lower and upper bound of the U_i variables. The load on vehicle at departure from a node i can be at least equal to the demand of that node and it can at most be equal to the capacity of the vehicle. Constraints (4.8) ensure that no demand is assigned to a location if that location is not selected (that is, $y_j = 0$) and if a candidate location is selected (that is, $y_j = 1$) then the sum of the demands that are assigned to the selected facility should not exceed the capacity of the facility that will be constructed on that location. Constraints (4.9) specify that customer i can be assigned to a location j only if that location is selected. Constraints (4.10) require that each customer is assigned to exactly one depot. Constraints (4.11) state that if a customer is not assigned to a location then there cannot be an arc that goes from that customer to that location's arrival node. Similarly, constraints (4.12) state that if a customer is not assigned to a location then there cannot be an arc that goes from the departure node of that location to this customer. Constraints (4.13) and (4.14) ensure that if customer j follows customer i and if customer i is assigned to candidate location k then customer j must also be allocated on that same candidate location. The remaining constraints (4.15), (4.16) and (4.17) specify that the routing variables (x_{ij}), the allocation variables (z_{ik}) and the location variables (y_j) can take values 0 or 1.

The given exact model is observed to solve small size LRPs (up to 15 customers and 4 depots) optimally in the computational experiments that we conducted. Hence, we propose a column generation algorithm tailored to the solution of the LRP. The algorithm is presented in section 4.3 in detail.

4.3 Column Generation Algorithm

4.3.1 The Master Problem

A feasible solution of a LRP is composed of the selected subset of locations and a set of feasible routes outgoing from these selected locations. A route is declared as feasible if it does not violate vehicle capacity restrictions, if each customer assigned to the route is

visited exactly once (it means that the route should be an elementary route) and if the route starts and ends at the same depot. The set of feasible routes that can exist in the optimal solution of LRP is considered while constructing the LRP master problem. Initially, if we assume that we have all feasible routes then the LRP reduces to selecting a subset of these feasible columns such that the cost of the routes and fixed cost of opening the facilities that these routes originate are minimized. Each feasible route forms a binary variable (that is, a column) of the MP model. The MP model solves the LRP if all the feasible columns, the set R, is readily available. Given the set R, the task of the MP model reduces to assign all customers to the routes such that each customer is assigned to exactly one route and MP also handles the facility capacity constraints so that feasibility is not violated. That is why, the MP is a set partitioning problem with side constraints. The natural issues that come to mind are the difficulty of finding the feasible routes efficiently and the exponential number of feasible routes that should be considered. The problem of huge number of feasible routes is addressed by relaxing the master problem which is an integer programming problem and solving the resultant linear program (RMP) by column generation. We start the column generation algorithm by considering only a subset of feasible columns. Then using the insight of solving a linear programming (LP) problem, we add only the columns that have negative reduced costs to the master problem. The problem of finding the feasible routes that have negative reduced costs is solved in the subproblem of the column generation algorithm and explained in the section 4.3.2. The mathematical formulation of the master problem is given below:

Sets:

R, set of feasible routes

$J = \{1, 2, .., M\}$, set of candidate locations

$C = \{1, 2, .., N\}$, set of customers

Model Parameters:

K: Number of available vehicles

δ_{ir}: 1 if the customer i is assigned to route r, 0 otherwise; $i \,\epsilon\, C, r \,\epsilon\, R$

μ_{jr}: 1 if the route r originates from facility j, 0 otherwise; $j \,\epsilon\, J, r \,\epsilon\, R$

PD_r: Demand of route r; $r \,\epsilon\, R$

C_r: Cost of route r; $r \,\epsilon\, R$

T_j: Capacity of candidate facility j; $j \epsilon J$

FC_j: Fixed cost of candidate facility j; $j \epsilon J$

Decision Variables:

x_r: 1 if the route r is selected, 0 otherwise; $r \epsilon R$

y_j: 1 if the candidate facility j is selected, 0 otherwise; $j \epsilon J$

Minimize

$$\sum_{r \epsilon R} C_r x_r + \sum_{j \epsilon J} FC_j y_j \tag{4.18}$$

subject to

$$\sum_{r \epsilon R} \delta_{ir} x_r = 1, \qquad \forall i \epsilon C, \tag{4.19}$$

$$y_j \leq \sum_{r \epsilon R} \mu_{jr} x_r \leq K y_j, \qquad \forall j \epsilon J, \tag{4.20}$$

$$\sum_{r \epsilon R} \delta_{ir} \mu_{jr} x_r \leq y_j, \qquad \forall i \epsilon C, \ \forall j \epsilon J, \tag{4.21}$$

$$\sum_{j \epsilon J} y_j \geq LB_w, \tag{4.22}$$

$$LB_v \leq \sum_{r \epsilon R} x_r \leq K, \tag{4.23}$$

$$\sum_{r \epsilon R} PD_r \mu_{jr} x_r \leq T_j y_j, \qquad \forall j \epsilon J, \tag{4.24}$$

$$x_r \in \{0,1\}, \qquad \forall r \epsilon R \tag{4.25}$$

$$y_j \in \{0,1\}, \qquad \forall j \epsilon J \tag{4.26}$$

The objective function of the MP is to minimize the cost of the selected routes among the set of feasible routes R and the fixed cost of selected facility locations. Objective function values of the MP correspond to the exact LRP model presented previously. Note that the variable costs for each selected location is incorporated to the cost of each route found. Hence, the cost of a new route $r = \{i_1, i_2, .., i_t\}$ can be found by the equation (4.27):

$$C_r = CM \sum_{s=1}^{s=t-1} d_{i_s i_{s+1}} + VC_{i_1} \sum_{s=2}^{s=t-1} q_s \tag{4.27}$$

Note that in route r, i_1 is the departure node of a candidate location and i_t is the corresponding arrival node of this candidate location. We add the demand of the customer nodes $\{i_2, .., i_{t-1}\}$ only in the second term of the formula and obtain the route demand and

multiply it by the variable cost of the candidate location i_1. Remember that variable cost is the cost per unit throughput from the given location.

Constraints (4.19) are defined for each customer and form the set partitioning constraints. These constraints state that each customer i is served by exactly one route among the set of feasible routes R. Constraint set (4.20) restricts the number of paths originating from a candidate facility location. If a candidate location j is selected then it requires that at least one route originates from that location and the maximum number of routes that can originate from the location is set to the available number of vehicles. If the candidate location j is not selected then it requires that no routes will originate from this location. Constraint set (4.21) is first given in [5]. One classical constraint that can be written instead of this constraint could be $x_r \leq \sum_{j \epsilon J} \mu_{jr} y_j, \ \forall r \, \epsilon \, R$. The classical one is written for each route r and simply states that if a route r that starts at facility j is selected, then facility j must also be selected. Note that the right term corresponds to the facility from which route r originates. For the preferred constraint set let R_{ij} denote the set of routes that includes customer i and depot j pair simultaneously. It is clear that the sum of route selection variables over the set R_{ij} must be at most one since only one route that includes the same $i - j$ pair can be selected in a feasible solution. So this can be written by the following constraint:

$$\sum_{r \epsilon R_{ij}} x_r \leq y_j, \qquad \forall i \, \epsilon \, C, \ \forall j \, \epsilon \, J \tag{4.28}$$

Note that the left side of the constraint (4.28) is tighter when compared to the classical one. Therefore, relaxation values of the model with constraint set (4.28) will be greater than or equal to relaxation values with the classical model. Another advantage is the reduced number of constraints since (4.28) is defined for each customer and candidate location. However, the classical constraint is defined for each feasible route R. Note that by using the δ_{ir} and μ_{jr} parameters, (4.28) can be rephrased as the constraint (4.21), that is, they are equivalent.

Constraint (4.22) sets the lower bound on the number of facilities that can be selected among the candidate set. LB_w is calculated as $\lceil \sum_{i \in C} d_i / T_{max} \rceil$ where T_{max} is the maximum candidate facility capacity among all. Constraint (4.23) states that the total number of routes selected (or vehicles used) should not exceed the the number of available vehicles

and it should be greater than LB_v which is calculated as $\lceil \sum_{i \in C} d_i/Q \rceil$. Constraint set (4.24) requires that sum of the demands of the customers assigned to a facility should not exceed the facility's capacity if the facility is selected. The constraints (4.25) and (4.26) specify that the route selection variables (x_r) and the location variables (y_j) can take values 0 or 1.

4.3.2 The Pricing Problem

The *promising* feasible columns are generated in the subproblem of the column generation and added to the master problem. In a simplex algorithm for a minimization problem, a promising nonbasic variable is the one with the negative reduced cost. The reduced cost defines the reduction in the objective function if the nonbasic variable is increased by one unit. Recall from the simplex algorithm that the reduced cost of a variable x_r is given by $\hat{c}_r = c_r - c_B B^{-1} A_r$. In this notation A_r is the r^{th} column corresponding to the variable x_r and c_r is the objective function coefficient of the variable x_r. Note that $c_B B^{-1}$ gives the dual values (π) of the solution of the LP model. So given the dual values obtained from the solution of the relaxed master problem (RMP), the objective of the subproblem is to find a new feasible column such that reduced cost of the column is negative, that is the column is promising. The new columns identified in the subproblem are added to the set R, prepared for the RMP and then RMP model is resolved. Column generation algorithm is stopped when the pricing subproblem cannot identify a feasible route with a negative reduced cost.

We can identify promising columns by solving an elementary shortest path problem with resource constraints (ESPPRC) for each candidate location j. The subproblem is considered on a directed subgraph $G^j = (V^j, A^j)$, where A^j is the set of arcs and $V^j = \{0, 1, ..., N, N+1\}$ is the set of nodes. For the candidate location j, the departure and arrival nodes are shown by the nodes 0 and $N+1$, respectively. A typical route will start from node 0 and end with node $N+1$. An important point is the cost of paths in the subproblem for the network constructed for a facility location j; the cost of any path found must correspond to the reduced cost of the path variable in the master problem. In other words, the sum of the arc costs of a route identified in a subproblem must give the reduced cost of that route if it had been in the MP model with the current dual values. Table 4.1 shows the constraint number and associated dual values that are obtained as the master problem is solved. Thus,

MP constraint number	Dual value
4.19	$\pi_i^{4.19} \quad \forall i \, \epsilon \, C$
4.20	$\pi_j^{4.20} \quad \forall j \, \epsilon \, J$
4.21	$\pi_i^{4.21} \quad \forall i \, \epsilon \, C$
4.23	$\pi^{4.23}$
4.24	$\pi_j^{4.24} \quad \forall j \, \epsilon \, J$

Table 4.1: Constraints and associated dual values for the Master Problem

the modified arc cost $\hat{c}_{ij}$ in the constructed network of the subproblem is given by;

$$\hat{c}_{ij} = \begin{cases} c_{ij} - \pi_i^{4.19} + \pi_i^{4.21} - \pi_0^{4.24} * q[i] & \text{if } i \, \epsilon \text{ C and } j \, \epsilon \text{ C} \cup \text{N+1} \\ c_{ij} - \pi_0^{4.20} - \pi^{4.23} & \text{if } i = 0 \text{ and } j \, \epsilon \text{ C} \end{cases}$$

where

$$c_{ij} = \begin{cases} CM * d_{ij} + q_i * VC_0 & \text{if } i \, \epsilon \text{ C and } j \, \epsilon \text{ C} \cup \text{N+1} \\ CM * d_{ij} & \text{if } i = 0 \text{ and } j \, \epsilon \text{ C} \end{cases}$$

Note that modified arc costs are calculated only if the corresponding arc variables are defined, otherwise they are set to infinity.

Based on our observations in the computational experiments, one of the most critical issues to create a time-efficient algorithm is to solve the subproblem effectively. The subproblem which is an ESPPRC was shown to be strongly NP-Hard by Dror [17]. An integer model whose solution identifies an elementary shortest path on the defined subnetwork for a location j would give only the column with the most negative reduced cost. However, the goal is to determine whether a negative reduced cost column exists or not. So obtaining a set of good routes for each location j is more important than finding the best route for a fast evolving algorithm. That is why, for the pricing subproblem we use dynamic programming to find a set of routes that have negative reduced costs. The structure of the algorithm we used for solving the ESPPRC is presented in Figure 4.2. A path is called non-elementary if a node is visited more than once hence if the path contains any loop; otherwise it is called an elementary path. The dynamic programming algorithm proposed solves elementary shortest path problem with resource constraints by modifying the algorithm of Larsen [49] that solves the relaxed version of the ESPPRC in which non-elementary paths are allowed. In

his dynamic programming algorithm, Larsen [49] uses the basic principles of Dijkstra's algorithm in order to find feasible routes with negative reduced costs. Normally, an optimal solution to a LRP does not involve cycles since each customer is visited once. However, cycle constraint is generally relaxed and a shortest path problem with resource constraints (SPPRC) is solved. SPPRC is also NP-Hard but pseudo-polynomial algorithms exist to solve the SPPRC [15] hence they can be solved more efficiently (e.g. see [1]). Using this solution approach, the solution of the RMP might contain non-elementary paths since non-elementary paths are added to the MP. However, it has been shown that an optimal integer solution will only contain elementary paths because of the triangular inequality assumption [8]. The disadvantage of this approach is that the lower bound obtained by the solution of RMP is weakened since negative cycles help to reduce costs. In our dynamic programming algorithm we store the predecessor nodes for each partial path and eliminate k-cycles.

The modified arc costs are updated each time RMP is solved. We create the initial label {0,0} in step 1 of the dynamic programming algorithm. Each label ($\{i, AccDem_i\}$) represents a partial path. The first entry in a label is the last node of the partial path and the second is the accumulated demand for the partial path. Cost of a label is shown by $Cost(\{i, AccDem_i\})$. The cost of the initial label is set to 0. When a label is created it is added to the set of unprocessed labels. In step 2, $BestLabel()$ function chooses the label that has minimum accumulated demand among the unprocessed labels. Since the labels are processed in order of increasing accumulated demand, a label can be chosen and processed only one time in step 2. Note that the label is removed from the set of unprocessed labels at the end of step 2 after it has been processed. The label $\{u, AccDem_u\}$ that has the minimum accumulated demand is extended to the other nodes if the extension is feasible and the cost is improved. If a label could be created after the feasibility and improvement checks, it is added to the set of unprocessed labels. $CreateLabel()$ function checks whether the label ($\{v, AccDem_v\}$) that will be created as the result of extension already exists or not. If it exists the cost of old label is updated, otherwise the label is new and its cost is taken as infinite while doing the improvement check.

Dynamic Programming Procedure for solving ESPPRC

Inputs: Updated arc costs, demand of each node and vehicle capacity

Output: A set of elementary paths from node 0 to node $N+1$ that have negative reduced costs and do not violate any capacity constraints.

Step 1. Initialization

Get the updated arc costs associated with the last RMP solved

CreateLabel({0,0})

Initialize a set of sets that keep elements of each distinct path found

Initialize predecessor set

Step 2. WHILE $u \neq N+1$ **DO**

$\{u, AccDem_u\}$ =BestLabel(unprocessed labels)

IF $u < N+1$ **THEN**

FOR each $v \epsilon\ C \cup \{N+1\}$ **DO**

Feasibility check:

IF $u \neq v$ **AND** $\hat{c}_{uv} \neq \infty$ **AND** $AccDem_u + d_v <= Q$ &

$v \notin$ preds[$\{u, AccDem_u\}$] **THEN**

$AccDem_v = AccDem_u + d_v$

Improvement check:

IF $Cost(\{u, AccDem_u\}) + \hat{c}_{uv} < Cost(\{v, AccDem_v\})$ **THEN**

isCreateLabel=DominanceRule()

IF $isCreateLabel = true$ **THEN**

$CreateLabel(\{v, AccDem_v\})$

preds[$\{v, AccDem_v\}$] =preds[$\{u, AccDem_u\}$] $\cup \{v\}$

IF $v = N+1$ **AND** $Cost(\{u, AccDem_u\} + \hat{c}_{uv} < 0$ **THEN**

Form a set from the nodes of the distinct path

IF The node set of the path is distinct **THEN**

Keep the order of the node set

ELSE IF The node set already exists **THEN**

Update the order of the node set

Remove label $\{u, AccDem_u\}$ from the set of unprocessed labels

Step 3. FOR each distinct path sets **DO**

Prepare the routes found for the MP

Figure 4.2: Dynamic Programming Algorithm for ESPPRC

In order to prevent visits to nodes more than once, we keep the predecessor nodes of each label in $CreateLabel()$ function and during the feasibility check we extend a label $\{u, AccDem_u\}$ to a node v only if node v is not an element of the partial path ending at node u. The improvement check is positive if the cost of the old label can be decreased or if the label is created for the first time. However, we do not create all the labels that can pass the improvement check. The reason is that there may be exponentially many partial paths that are feasible even for small size problems and in order to have a more efficient algorithm we may not create labels that are dominated by other labels. The basic dominance used for solving the non-elementary shortest path problem with resource constraints is as follows: Let $\{i, AccDem_1\}$ and $\{i, AccDem_2\}$ be two partial paths ending at the same node i. Label $\{i, AccDem_1\}$ dominates label $\{i, AccDem_2\}$ only if the following conditions are satisfied:

$$Cost(\{i, AccDem_1\}) < Cost(\{i, AccDem_2\}) \tag{4.29}$$

$$AccDem_1 \leq AccDem_2 \tag{4.30}$$

If an extension of the second path is feasible then the same extension will also be feasible for the first path with a lower cost considering the conditions given in Eqs. (4.29) and (4.30). Thus, the label for the second path can be discarded. However, for the elementary shortest path problem with resource constraints the basic dominance rule cannot be applied directly and a modified dominance rule has been proposed first by Dumas and Desrosier [18] and Chabrier [8]. For SPPRC, the first and second path can both continue on the same node if vehicle capacity is not exceeded. However, for ESPPRC the second path has the chance of continuing with a node of the first path and this extension might be useful later. Note that the first path cannot continue with a node already present in its partial path. Hence, the promising node cannot be added to the first path. A third condition that is required for the first path to dominate the second is given in (4.31).

$$S_{\{i,AccDem_1\}} \subseteq S_{\{i,AccDem_2\}} \tag{4.31}$$

$S_{\{i,AccDem_1\}}$ denotes the set of nodes of the first partial path. The third condition requires that the set of nodes of the first path is included in the second path so that possible successors of both paths are the same. Similar to [8], our computational experience also shows that if all the three conditions are applied, the number of labels discarded due to dominance are much less compared to basic dominance. Hence, our strategy is to apply

basic dominance as long as the algorithm could find columns with negative costs. When the algorithm cannot identify a column with negative cost, the third condition is also applied to decide on dominance. As the algorithm continues, if a label $\{N+1, AccDem_v\}$ with negative reduced cost is discovered, the set $S_{\{N+1,AccDem_v\}}$ is formed from the nodes of the label. Let S_{N+1} denotes the set that keeps sets of each distinct path as its elements. If $S_{\{N+1,AccDem_v\}} \in S_{N+1}$, then the path is already discovered and we only update the order of the nodes. But if $S_{\{N+1,AccDem_v\}} \notin S_{N+1}$, then this set is a newly discovered distinct path and we add the path to the set S_{N+1}. The solution of the subproblem for facility location j is terminated when the unprocessed labels are only the ones associated with $N+1$. The labels $\{N+1, AccDem_v\}$ are not processed since node $N+1$ is the arrival node and hence they cannot be extended to a further node. If $BestLabel()$ function must return a label of node $N+1$ since all the other labels are processed then the algorithm continues with step 3. In this step, all the distinct paths associated with node $N+1$ are prepared for and added to the master problem.

4.3.3 The Algorithm

Column generation is an effective method to solve large-scale linear programming problems. Our work proposes a column generation algorithm for the solution of capacitated location-routing problem (The vehicles and facilities are capacitated). The master problem (MP) in our column generation algorithm is a set-partition based formulation which proved to be useful for a variety of routing problems such as the vehicle routing problem with time windows. The MP model is an integer programming model and is explained in detail in the section 4.3.1. Column generation technique is particularly applied for the solution of the relaxed master problem (RMP) which happens to have an exponential number of feasible columns. The RMP is the LP relaxation of the MP. As in any column generation technique, we start with a subset of feasible columns and we try to find new promising columns in the subproblem which is also called the pricing problem. For the pricing problem we solve elementary shortest path problem with resource constraints by modifying the algorithm of Larsen [49]. The objective function value of the RMP forms a lower bound to the original problem when no more feasible routes with negative reduced cost can be found by the subproblem. The outline of our column generation algorithm is given in Figure 4.3.

Steps of Column Generation Procedure

Inputs: Coordinates and demand of each node,

Variable cost, fixed cost and capacity for each candidate location

Available number of vehicles, vehicle capacity and cost per mile

Output: An integer solution and a gap for the given LRP

Step 1. Initialization

1.1 Get the inputs

1.2 Create departure depot(D) and arrival depot(A) sets

1.3 Calculate distance matrix

1.4 Initialize routes

Construct single-node routes

Construct multi-node routes using heuristics

1.5 Prepare the initial set of routes for the Master Problem (MP)

Step 2. Construct Master Problem

Solve Integer Master Problem with the initial set of routes

Step 3. WHILE $RDcost_r < 0$ for any new path r **DO**

3.1 Solve Relaxed Master Problem (RMP)

Update the lower bound

Get the dual values

Modify the edge costs using the dual values

IF the solution is not fractional

OR gap $\geq 3\%$ (checked at regular intervals) **THEN**

Solve Integer Master Problem

Update the upper bound

Keep the solution as the starting feasible solution

3.2 FOR each $j \epsilon J$ **DO**

Solve Subproblem with modified costs

Prepare the paths with negative reduced cost for the MP

Figure 4.3: Column Generation Algorithm

In the initialization step, we first import the data for the given problem. After determining the number of customers (N) and the number of candidate locations (M), the set of departure locations (D) and the set of arrival locations (A) are established (Step 1.2). The arc travel costs are calculated by multiplying the Euclidean distances between pairs of coordinates with the cost per mile parameter (CM). The distances are calculated with five decimal point and truncation with the formula given in equation (4.32).

$$d_{ij} = \frac{\lfloor (10^5)\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor}{10^5} \tag{4.32}$$

The column generation algorithm starts with an initial set of feasible columns which form a feasible solution. We could use the single-node routes, that is, we could assign each customer to each depot and have $N * M$ starting routes or columns. Note that each feasible route will be a variable or equivalently a column of the master problem. However, this approach has some disadvantages. First, since we would have one customer in each route, we would have N routes for the initial feasible solution. Therefore the number of available vehicles (K) that restricts the number of routes to be used could not be incorporated in the master problem model in this case. Constraints (4.20) and (4.23) of the MP model must be removed to have a feasible solution and the relaxation gap increases if they are not utilized; therefore, the integer MP is solved in a longer computing time. Second, the nature of the LRP requires to fill the vehicle capacity as much as possible to force smaller number of depots and vehicles. That is why, multi-node routes that utilize the vehicle capacity better are likely to be present in optimal solutions. As expected, using single-node routes only gives worse initial feasible solutions in the sense that they are far from optimal. Hence, the number of pricing problems that should be solved to find good quality routes increases. To construct an initial set of multi-node routes, we apply a heuristic method for each candidate depot j. The routes are found by using Clarke and Wright Savings algorithm [10] as a construction heuristic and it is followed by a series of route improvement heuristics, namely 2-opt, 1-1 exchange, crossover and simulated annealing as in the case of VRPTW.

In step 1.5, the initial set of feasible routes are prepared for the master problem. Note that each feasible route r is incorporated to the master problem as a variable x_r, that is why, in order to prepare a route for the MP, the modelling components related to the variable x_r must be constructed. Hence, in the preparation step, we calculate the cost of the route (C_r),

the demand of the route (PD_r) and arrange the parameter δ_{ir} which holds the customers assigned to route r and parameter μ_{jr} which holds the depot that is used in route r. The parameter δ_{ir} is 1 if the customer i is assigned to route r, 0 otherwise and the parameter μ_{jr} is 1 if the route r originates from depot j, 0 otherwise. In step 2, we construct the MP model which is an integer programming problem and when we solve the master problem for the first time with these prepared columns, we obtain an initial feasible solution and keep the initial solution to serve as a starting integer solution for our instance. Note that by using CPLEX 9.1, MP model is constructed columnwise only one time and later the newly found columns are added to the model without reconstructing the entire model but by using the $addColumn()$ routine of the CPLEX.

We iterate step 3 until we cannot find a feasible column that has negative reduced cost for any candidate location j. After the relaxed master problem (RMP) is solved (step 3.1), the lower bound is updated and the edge costs are modified using the dual values. At this moment, it is very beneficial to check whether the solution of the RMP is integral or not. Computational experience showed that it is not uncommon to have integral solutions for the RMP since the size of the RMP increases step by step at each iteration of step 3.2. Especially for the initial iterations in which small size LPs are solved, feasible region is defined well for the given MP. Therefore, it is more difficult for a fractional solution to be found. If an integral solution is found for the RMP, this integral solution is clearly the optimum of the integer MP problem for the feasible set of columns R at that MP instance. This means that solving the integer problem with the given columns at hand would finish at node 0 of the branch-and-bound algorithm and hence it would be very easy to update the upper bound if the LP solution is integral. Frequent updates of the upper bound and keeping the last upper bound found as the starting initial solution for the integer MP is one of the important strategies applied in our algorithm. The goal is to keep the gap between the upper bound and the lower bound as small as possible during the execution of the algorithm. If the only upper bound kept would be the one we get at step 2 from the solution of the first integer master problem with the initial set of routes, then closing the gap when the loop at step 3 ends would be more difficult. Note that column generation is used to solve the RMP. So the lower bound at hand at the end of the algorithm would be the first relaxation of the integer MP problem while applying the branch-and-bound. It is clear that the smaller the

gap is at node 0 of the branch-and-bound tree, the faster the branch-and-bound execution is going to be. To keep the gap manageable, we solve an integer problem after each 20 RMP solutions if the gap calculated at that instance is greater than 3%. In step 3.2, we solve the subproblem for each candidate location j and identify the columns that will be added to the master problem and hence used in the solution of step 3.1. We restrict the maximum number of columns to be added for a single candidate location to 50 columns. Hence a maximum of $50 * M$ columns are added in each solution of step 3.2.

The algorithm continues until no more feasible routes with negative reduced cost are discovered at step 3 and the objective value of the RMP at this stage specifies the lower bound for the integer problem.

4.4 Implementation Details

4.4.1 Test Problems

We try to solve a general LRP where we have capacitated multiple facilities and multiple vehicles. There are no commonly accepted and widely studied benchmark problems for this type of LRP. The 3 benchmark problems of Perl [59] are the only problem sets that match with the general LRP we study. LRP benchmarks of Perl [59] are studied in [60], [29] and [73]. The authors of these papers compare their computational results with each other. Another LRP data set which is close to the general LRP we study is provided by Or [57] and is studied in [58]. However, facilities are uncapacitated in the LRP benchmark of Or [57]. Recently, Barreto [3] tried to form a standardized database for the general LRPs by respecting the format proposed by Perl [59]. The instances that are compiled by Barreto [3] are mostly from the literature of vehicle routing problems. The seed papers or theses for these instances are due to Perl [59], Daskin [13], Gaskell [23], Min [55], Or [57], Srivastava [64] and Christofides [9]. In order to form a standardized database Barreto took the depot variable costs as zero and take the cost per mile parameter as 1 in all the instances. However, while making computational tests we respected the original data of Perl's 3 benchmarks. Therefore, we could make comparisons with the solutions of [60], [29] and [73]. For the other instances, we could compare our results only with [4].

4.4.2 Computational Platform

Computational experiments are performed on a HP xw9300 Workstation with an AMD Opteron Processor 2.59 GHz and 8.00 GB of RAM at the Systems Lab of Koç University, Istanbul. We report the total elapsed times, not the CPU time for each instance.

We have used ILOG CPLEX version 9.1 [30]. Mixed integer and primal simplex optimizers of CPLEX are used to solve MIP and LP models respectively. The algorithm is implemented with Java API of ILOG CPLEX Concert Technology.

4.4.3 Computational Results

Performance of the column generation algorithm has been tested on the 18 benchmark instances. Names of the instances are formed by the name of the researcher, number of customers and number of candidate facility locations. For instance, Perl-55c-15d refers to the Perl's benchmark problem with 55 customers and 15 candidate facility locations. The results obtained for the instances are presented in Table 4.2. In this detailed table, vehicle capacity (Vcap), integer objective function value (IP), lower bound obtained from the column generation procedure (LB), integrality gap (Gap), number of vehicles/routes used (Vno) and the set of locations selected (Depots) in the solution and finally the runtime of the column generation algorithm (Time(s)) is given respectively. More detailed solution information that includes the routes and the cost of each route obtained for each instance is also presented at the appendix in Table A.1.

All the studies that we compare our results are heuristic methods. Unfortunately, we do not have a peer exact method to infer on our solution efficiency. Note that in a benchmark analysis, even if the methods are all heuristic methods or all exact methods, it would still be difficult to compare the run times that are taken across different computational platforms. We reported our own running times in Table 4.2. We solved 11 out of 18 instances within 1 hour and 13 instances within 4 hours. The generation of promising columns continued until an elapsed time limit of 12 hours for 5 of the instances (Christofides-100c-10d, Min-134c-8d, Daskin-88c-8d, Daskin-150c-10d, Or-117c-14d). For these instances, the final integer solution is reported and TILIM is written in the runtime column to indicate that algorithm has been terminated due to time limit.

For the Perl's 3 benchmark problems (Perl-12c-2d, Perl-55c-15d and Perl-85c-7d), we

Table 4.2: Results on the LRP Benchmark Instances

No	Problem Name	Vcap	IP	LB	Gap (%)	Vno	Depots	Time(sec)
1	Gaskell-21c-5d	6000	424.899	424.899	0.00	4	{1,2}	13.39
2	Gaskell-22c-5d	4500	585.108	585.108	0.00	3	{1}	128.25
3	Gaskell-29c-5d	4500	512.103	504.965	1.39	4	{2,3}	468.80
4	Gaskell-32c-5d	8000	568.836	544.316	4.31	4	{3}	1233.00
5	Gaskell-32c-5d	11000	504.329	502.396	0.38	3	{3}	2576.16
6	Gaskell-36c-5d	250	460.374	456.257	0.89	4	{5}	180.98
7	Christofides-50c-5d	160	565.618	562.261	0.59	5	{2,5}	3235.09
8	Christofides-75c-10d	140	867.326	835.855	3.63	12	{1,5,9}	11027.27
9	Christofides-100c-10d	200	849.468	-	-	9	{2,8}	TILIM
10	Perl-12c-2d	140	355.582	355.582	0.00	2	{1}	1.16
11	Perl-55c-15d	120	5465.510	5400.751	1.18	10	{2,8,12}	284.61
12	Perl-85c-7d	160	7496.618	7379.341	1.56	11	{2,4,6}	6778.31
13	Min-27c-5d	2500	3062.017	3062.017	0.00	4	{2,3}	105.34
14	Min-134c-8d	850	6287.440	-	-	13	{2,3,4,5}	TILIM
15	Daskin-88c-8d	9000000	431.080	-	-	7	{5,7}	TILIM
16	Daskin-150c-10d	8000000	50234.162	-	-	12	{1,2,9}	TILIM
17	Or-117c-14d	150	13193.491	-	-	6	{1,2,5}	TILIM
18	Srivastava-8c-2d	450	472.665	472.665	0.00	2	{1,2}	2.59
			Average Gap		1.07			
			Median		0.59			

compare our results with the results of Perl [59], Hansen et al. [29] and Wu et al. [73]. All the 3 methods are heuristic methods as discussed in the literature. The results are summarized in Table 4.3. It is observed that the solution found for the Perl-12c-2d is the same as the heuristic solutions and the solutions found for Perl-55c-15d and Perl-85c-7d is lower than all the previous results.

In Table 4.4, we compare the results reported in Barreto et al. [4] with our results. In the table provided by Barreto et al. [4], the (IP) column is the best known solution obtained using the clustering heuristic. The lower bound (LB) column was obtained by Barreto [3] with a relaxed 2-index integer linear programming formulation. It was not possible to get LB for the instance Min-134c-8d using the relaxed 2-index integer linear programming formulation. Using the column generation algorithm, a valid LB could not be found in 5 of the instances since the algorithm terminated due to time limit, meaning that the algorithm was continuing to generate promising columns when the time limit has been reached. The

	Perl-12c-2d			Perl-55c-15d			Perl-85c-7d		
	Depots	Vno	Costs	Depots	Vno	Costs	Depots	Vno	Costs
Perl [59]	{1}	2	355.58	{2,10,12}	10	5795.62	{2,4,5}	11	7789.96
Hansen et al.[29]	{1}	2	355.58	{2,7,12,13}	10	5617.67	{2,4,6}	11	7551.61
Wu et al. [73]	{1}	2	355.58	{5,10,12}	10	5532.28	{2,4,6}	12	7781.21
Proposed Algorithm	{1}	2	355.58	{2,8,12}	10	5465.51	{2,4,6}	11	7496.62

Table 4.3: Comparison on Perl's Benchmark Problems

optimality is reached in 2 of the 14 instances (Gaskell-29c-5d and Min-27c-5d) by using the clustering heuristic method. In 8 of the 14 instances the proposed integer solutions is lower than the Barreto's solutions. The solutions obtained in both of the methods are the same in 2 of the 14 instances. In the remaining 4 instances Barreto obtained better results compared to our solutions. In all these 4 instances our algorithm terminated due to time limit. For the instance Christofides-100c-10d, although the proposed algorithm terminated due to time limit, the solution found is better than the Barreto's solution. Barreto et al. [4] reports that the integrality gap (Gap %) falls between a minimum of 0% and a maximum of 19.01% with an average of 4.81% and a median of 3.13% (In this calculation the instances of Perl are also added). For the same 9 instances solved by both of the algorithms in Table 4.4, the average and the median of gap of Barreto et al. [4] is 3.90% and 2.24% respectively. For the proposed algorithm, the integrality gap (Gap %) falls between a minimum of 0% and a maximum of 4.31% with an average of 1.24% and a median of 0.59%.

		Barreto et al. [4]			**Proposed Algorithm**		
Problem Name	**Vcap**	**IP**	**LB**	**Gap (%)**	**IP**	**LB**	**Gap (%)**
Gaskell-21c-5d	6000	435.9	424.9	2.59	424.9	424.9	0.00
Gaskell-22c-5d	4500	591.5	585.1	1.09	585.1	585.1	0.00
Gaskell-29c-5d	4500	512.1	512.1	0.00	512.1	505.0	1.39
Gaskell-32c-5d	8000	571.7	556.5	2.73	568.8	544.3	4.31
Gaskell-32c-5d	11000	511.4	504.3	1.41	504.3	502.4	0.38
Gaskell-36c-5d	250	470.7	460.4	2.24	460.4	456.3	0.89
Christofides-50c-5d	160	582.7	549.4	6.06	565.6	562.3	0.59
Christofides-75c-10d	140	886.3	744.7	19.01	867.3	835.9	3.63
Christofides-100c-10d	200	889.4	788.6	12.78	849.5	-	-
Min-27c-5d	2500	3062	3062	0.00	3062.0	3062.0	0.00
Min-134c-8d	850	6238	-	-	6287.4	-	-
Daskin-88c-8d	9000000	384.9	356.4	8.00	431.1	-	-
Daskin-150c-10d	8000000	46642.7	43938.6	6.15	50234.2	-	-
Or-117c-14d	150	12474.2	12048.4	3.53	13193.5	-	-
		Average Gap		3.90	**Average Gap**		1.24
		Median		2.24	**Median**		0.59

Table 4.4: Comparison on Benchmark Problems that are compiled by Barreto [3]

Chapter 5

CONCLUSIONS

In this thesis we studied two well-known routing problems. The first problem is the Vehicle Routing Problem with Time Windows (VRPTW) and the second one is the location-routing problem (LRP). For the VRPTW, we used a two-index model and applied a branch-and-cut algorithm. Previously, exact algorithms that use column generation to solve the shortest path decompositions of the problem gave the best results [14, 49, 11, 8]. However, we were motivated by the promising results of Bard et al. [2] who applied a branch-and-cut procedure in order to minimize the total number of vehicles but not the total distance traveled. While designing our branch-and-cut algorithm, we have introduced a new strategy that aims to close the relaxation gaps quickly. We applied a separation routine that detects all cycles in the nodes of the branch-and-cut tree in polynomial time and tried to eliminate these cycles by checking the violated valid inequalities. A new branching strategy that branches on the cycles found in the node relaxations of the MILP model has also been presented. It is observed that the branch-and-cut algorithm performs considerably well especially on the long horizon test problems. Compared to leading algorithms in the literature, average and median of solution times for the same long horizon problems are significantly better for the proposed algorithm. For most of the problems that could be solved to optimality, running times is less than 10 secs (62 out of 96 optimal results). For 94 instances out of 96 optimal results, the running times is under 30 minutes (The average running times for the 94 instances is 108.15 secs). Even for some harder instances that are newly closed in 2005 we could get better performance using our algorithm. The conclusion is that column generation based algorithms do not perform the best for all the Solomon test problems. The solution efficiency of the algorithm presented outperformed the application of a column generation for the long horizon instances. For most of the presented instances, running a pure branch-and-cut algorithm would be adequate, but for the larger instances column generation may be effective and achieved to solve previously unsolved instances.

For the Location-Routing Problem, we first introduced a formal MILP model where we have multiple capacitated vehicles and facilities. Then we applied a column generation algorithm tailored to the solution of this kind of LRP. This research is the first study that proposes a column generation algorithm for the solution of the general LRP that has capacitated multiple facilities and vehicles. The decomposition technique that is successfully applied in the vehicle routing problems has been utilized. The LRP has been decomposed into two subproblems. The master problem is a set partitioning problem with side constraints and the subproblem is an elementary shortest path problem with resource constraints. Column generation technique has been applied to solve the relaxed master problem (RMP) and a new dynamic programming algorithm has been designed in order to generate the columns in the subproblems. The gaps obtained are tightened since we solve an elementary shortest path problem hence eliminate all the k-cycles. It is observed that linear master problem is solved easily. Although the running times for the master problem increase as the number of columns increase, the total running times spent in solving the relaxed master problems is always negligible compared to solution times required for the subproblems. Hence, solving the subproblems efficiently is very crucial for a good column generation algorithm. The column generation algorithm has been successful in solving instances up to 10 candidate facility locations and 100 customers. For the larger instances the number of labels created at each solution of the subproblem has been very large and the running times to find a set of good routes increased. Therefore, the algorithm ran till the time limit for 5 of the 18 instances. Further work for the early elimination of infeasible or unnecessary labels is required to solve larger instances efficiently. For instance, adding a more sophisticated dominance criterion should improve the efficiency of the dynamic programming procedure hence affect the overall performance of the column generation algorithm. Future work should also focus on adding cutting planes to the master problem to allow larger instances to be solved.

During the execution of the column generation algorithm we focused to update bounds frequently and keep the gaps between the upper and lower bounds in a manageable level. It is confirmed in the literature that it is very crucial to have small gaps to solve the integer programs efficiently and computational experiments revealed that the strategy applied in the algorithm has resulted into small gaps. In our computational experiments, the integrality gap (Gap %) falls between a minimum of 0% and a maximum of 4.31% with an average of

1.18% and a median of 0.89%.

Benchmark problems are very important to check the solution quality and efficiency. Barreto's effort to set a widely accepted LRP benchmark problem sets is worthy. Benchmarks are crucial to make comparisons with the peer papers and it lets the researchers to test their solution quality and make inferences on the efficiency of their solution method. The number of papers that make comparisons with each other will probably increase for the general capacitated LRP as the benchmark problems for this kind of LRP are accepted and studied more in the near future.

As a future work, adding time window to the LRP problem will make it more close to the real life situations considering that some customers often impose service deadlines and desirable service time restrictions. The time window extension will be especially important with the advent of Just-In-Time (JIT) logistics principle. Min et al. [56] reports that only five LRP articles (less than 16%) account for hard time windows and none considers soft time windows. If a LRP with time windows is solved, the incompatible pair and path cuts that is introduced by Bard et al. [2] for the VRPTW can be incorporated to the master problem of the column generation algorithm.

BIBLIOGRAPHY

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] J.F. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36(2):250–269, 2002.

[3] S.S. Barreto. *Analysis and modelling of location-routing problems*. PhD thesis, Aveiro University, 2004.

[4] S.S. Barreto, C. Ferreira, J. Paixao, and B.S. Santos. Using clustering analysis in a capacitated location-routing problem. *European Journal of Operations Research (in print)*, 2006.

[5] R.T. Berger, C.R. Coullard, and M.S. Daskin. Location-routing problems with distance constraints. *Transportation Science*, 2006. Accepted for publication.

[6] L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and scheduling of vehicles and crews - the state of art. *Computers & Operations Research*, 10(2):62–212, 1983.

[7] J. Bracca, J. Bramel, B. Posner, and D. Simchi-Levi. A computerized approach to the new york city school bus routing problem. *IEE Transactions*, 29:693–702, 1997.

[8] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33:2972–2990, 2005.

[9] N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly*, 20(3):309–318, 1969.

[10] G. Clarke and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[11] W. Cook and J.L. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Department of Computational and Applied Mathematics, Rice University, 1999.

[12] G. Cornuejols, M.L. Fisher, and G.L. Nemhauser. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science*, 23(8):789–810, 1977.

[13] M.S. Daskin. *Network and discrete location: models, algorithms and applications*. John Wiley & Sons, Inc., New York, 1995.

[14] M. Desrochers, J. Desrosiers, and M.M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

[15] M. Desrochers and F. Soumis. A generalized permanent labelling algorithm for the shortest path problem with time windows. *INFOR*, 26(3):191–212, 1988.

[16] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Time constrained routing and scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Routing*, number 8 in Handbooks in OR/MS, chapter 2, pages 35–139. Elsevier, North-Holland, Amsterdam, 1995.

[17] M. Dror. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42:977–978, 1994.

[18] Y. Dumas and J. Desrosier. A shortest path problem for vehicle routing with pick-up, delivery, and time windows. Technical Report G-86-09, Les Cahiers du GERAD, 1986.

[19] S. Eilon, C.D.T. Watson-Gandy, and N. Christofides. *Distribution Management: Mathematical Modeling and Practical Analysis*. Hafner Publishing Company, Newyork, 1971.

[20] M. Fischetti. Facets of the asymmetric traveling salesman polytope. *Mathematics of Operations Research*, 16(1):42–56, 1991.

[21] M. Fisher. Vehicle routing. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Routing*, number 8 in Handbooks in OR/MS, chapter 1, pages 1–79. Elsevier, North-Holland, Amsterdam, 1997.

[22] R.W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.

[23] T.J. Gaskell. Bases for vehicle fleet scheduling. *Operational Research Quarterly*, 18(3):281–295, 1967.

[24] S. Gélinas, M. Desrochers, J. Desrosiers, and M.M. Solomon. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61:91–109, 1995.

[25] B.L. Golden, A.A. Assad, and E.A. Wasil. *Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries*, pages 245–286. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[26] B.L. Golden and E.A. Wasil. Computerized vehicle routing in the soft drink industry. *Operations Research*, 35:6–17, 1987.

[27] M. Grotschel and M. Padberg. Polyhedral theory. In Lawler, Lenstra, Rinooy Kan, and Shmoys, editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, chapter 2, pages 251–306. Wiley, Chichester, U.K., 1985.

[28] K. Halse. *Modeling and solving complex vehicle routing problems.* PhD thesis, IMSOR, Technical University of Denmark, Lyngby, 1992.

[29] P.H. Hansen, B. Hegedahl, S. Hjortkjaer, and B. Obel. A heuristic solution to the warehouse location-routing problem. *European Journal of Operational Research*, 76:111–27, 1994.

[30] ILOG, Inc. *ILOG CPLEX 9.1 User's Manual*, 2005.

[31] S. Irnich and D. Villeneuve. The shortest path problem with k-cycle elimination ($k \geq 3$): Improving a branch-and-price algorithm for the vrptw. Technical report, Lehr- und Forschungsgebiet Operations Research und Logistik Management, Rheinisch-Westfalische Technische Hochschule, 2003.

[32] S.K. Jacobsen and O.B.G. Madsen. A comparative study of heuristics for a two-level location-routing problem. *European Journal of Operational Research*, 5:378–387, 1980.

[33] B. Kallehauge and N. Boland. Path inequalities for the vehicle routing problem with time windows. Technical report, Centre for Traffic and Transport, Technical University of Denmark, 2005.

[34] Brian Kallehauge, J. Larsen, and O.B.G. Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers and Operations Research*, 33(5):1464–1487, 2006.

[35] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, chapter 2, pages 85–103. Plenum Press, New York, 1972.

[36] N. Kohl. *Exact methods for time constrained routing and scheduling problems*. PhD thesis, Institute of Mathematical Modeling, Technical University of Denmark, Lyngby, 1995.

[37] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis. k-path cuts for the vehicle routing problem with time windows. Technical Report IMM-REP-1997-12, Institute of Mathematical Modeling, Technical University of Denmark, Lyngby, 1997.

[38] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis. Two-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999.

[39] N. Kohl and O.B.G. Madsen. An optimization algorithm for the vehicle routing problem

with time windows based on lagrangean relaxation. *Operations Research*, 45:396–406, 1997.

[40] G. Kontoravdis and J.F. Bard. A grasp for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7(1):10–23, 1995.

[41] T. Kulcar. Optimizing solid waste collection in brussels. *European Journal of Operational Research*, 90(1):71–77, 1996.

[42] R.V. Kulkarni and P.R. Bhave. Integer programming formulations of vehicle routing problems. *European Journal of Operational Research*, 20:58–67, 1985.

[43] M. Labbe and G. Laporte. Maximizing user convenience and portal service efficiency in post box location. *Belgian Journal of Operations Research, Statistics and Computer Science*, 26:21–35, 1986.

[44] V. Lambert, G. Laporte, and F. Louveaux. Designing collection routes through bank branches. *Computers & Operations Research*, 20:783–791, 1993.

[45] G. Laporte. Location-routing problems. In B.L. Golden and A.A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 163–198. North-Holland Publishing, Amsterdam, Holland, 1988.

[46] G. Laporte and Y. Nobert. An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research*, 6:224–226, 1981.

[47] G. Laporte, Y. Nobert, and D. Arpin. An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research*, 6:293–310, 1986.

[48] G. Laporte, Y. Nobert, and P. Pelletier. Hamiltonian location problems. *European Journal of Operational Research*, 12:82–89, 1983.

[49] J. Larsen. *Parallelization of the Vehicle Routing Problem with Time Windows*. PhD thesis, Institute of Mathematical Modeling, Technical University of Denmark, Lyngby, 1999.

[50] J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227, 1981.

[51] O.B.G. Madsen. Methods for solving combined two level location-routing problems of realistic dimensions. *European Journal of Operational Research*, 12:295–301, 1983.

[52] R. Mechti, S. Poujade, C. Roucairol, and B. Lemarié. Global and local moves in tabu search: A real-life mail collecting application. *Manuscript, PriSM laboratory, University of Versailles, France*, 2001.

[53] J. Melechovský, C. Prins, and R. Wolfler Calvo. A metaheuristic to solve a location-routing problem with nonlinear costs. *Journal of Heuristics*, 11:375–391, 2005.

[54] C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulations and traveling salesman problems. *Journal of the ACM*, 7:326–329, 1960.

[55] H. Min, J. Current, and D. Schilling. The multiple depot vehicle routing problem with backhauling. *Journal of Business Logistics*, 13(1):259–288, 1992.

[56] H. Min, V. Jayaraman, and R. Srivastava. Combined location-routing problems: a synthesis and future research directions. *European Journal of Operational Research*, 108:1–15, 1998.

[57] I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Northwestern University, 1976.

[58] I. Or and W.P. Pierskalla. A transportation location-allocation model for regional blood banking. *AIIE Transactions*, 11:86–95, 1979.

[59] J. Perl. *A Unified Warehouse Location-Routing Analysis*. PhD thesis, Northwestern University, 1983.

[60] J. Perl and M.S. Daskin. A warehouse location-routing problem. *Transportation Research B*, 19B(5):381–396, 1985.

[61] G.K. Rand. Methodological choices in depot location studies. *Operational Research Quarterly*, 27:241–249, 1976.

[62] S. Salhi and G.K. Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 38:150–156, 1989.

[63] M.M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

[64] R. Srivastava. *Algorithms for solving the location-routing problem.* PhD thesis, The Ohio State University, 1986.

[65] R. Srivastava. Alternative solution procedures for the location-routing problem. *Omega International Journal of Management Science*, 21(4):497–506, 1993.

[66] R. Srivastava and W.C. Benton. The location-routing problem: Considerations in physical distribution system. *Computers & Operations Research*, 17(5):427–435, 1990.

[67] K. Tan, L. Lee, Q. Zhu, and K. Ou. Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering*, 15:281–295, 2001.

[68] P. Toth and D. Vigo. *The Vehicle Routing Problem.* SIAM Society for Industrial and Applied Mathematics, 2002.

[69] D. Tuzun and L. Burke. A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, 116(1):87–99, July 1999.

[70] M. Wasner and G. Zapfel. An integrated multi-depot hub-location vehicle routing model for network planning of parcel service. *International Journal of Production Economics*, 90:403–419, 2004.

[71] M.H.J. Webb. Cost functions in the location of depot for multiple delivery journeys. *Operational Research Quarterly*, 19:311–320, 1968.

[72] M.H.J. Webb. *Distribution Management: Mathematical Modeling and Practical Analysis*, volume 19. 1968.

[73] T-H. Wu, C. Low, and J-W. Bai. Heuristic solutions to multi-depot location-routing problems. *Computers & Operations Research*, 29:1393–1415, 2002.

Appendix A

DETAILED SOLUTIONS FOR THE LRP BENCHMARK PROBLEMS

Table A.1 lists the solutions found for the LRP instances. The routes found and the cost of each route (C_r) is reported for each instance. Note that C_r is calculated by the equation (4.27) which is given in Chapter 4 of the thesis.

Table A.1: The solutions to the LRP Benchmark Problems

Gaskell-21c-5d		
Routes		**C_r**
Route 1	22-19-21-20-17-27	59.447
Route 2	23-6-1-2-5-7-9-28	83.007
Route 3	23-10-13-11-4-3-8-28	95.547
Route 4	22-18-15-12-14-16-27	86.898
Gaskell-22c-5d		
Routes		**C_r**
Route 1	23-7-8-4-5-9-28	94.931
Route 2	23-13-10-28	48.749
Route 3	23-12-11-6-1-2-3-16-15-14-17-22-20-19-18-21-28	391.428
Gaskell-29c-5d		
Routes		**C_r**
Route 1	32-2-37	42.755
Route 2	32-4-5-1-6-24-25-29-27-28-26-3-37	173.048
Route 3	32-21-23-18-19-20-22-37	114.889
Route 4	31-13-16-15-10-11-12-8-14-9-17-7-36	81.410
Gaskell-32c-5d, Vcap=8000		
Routes		**C_r**
Route 1	35-30-31-2-12-11-1-40	92.415
Route 2	35-29-28-16-27-26-40	111.774
Route 3	35-3-4-5-6-7-9-8-10-32-13-15-40	174.233
Route 4	35-19-18-21-20-22-23-24-25-17-14-40	140.414
Gaskell-32c-5d, Vcap=11000		
Routes		**C_r**
Route 1	35-1-13-18-19-21-20-22-23-24-25-17-15-14-40	146.848
Route 2	35-30-29-28-16-27-26-40	129.676

Continued on next page

Table A.1 – continued from previous page

Route 3	35-31-12-2-3-4-5-6-7-8-9-10-32-11-40	177.804
Gaskell-36c-5d		
Routes		**C_r**
Route 1	41-21-27-26-25-31-32-33-34-28-22-46	109.174
Route 2	41-10-11-12-6-5-4-3-9-46	96.262
Route 3	41-23-29-35-36-30-24-18-17-16-46	106.068
Route 4	41-20-19-13-7-1-2-8-14-15-46	98.870
Christo-50c-5d		
Routes		**C_r**
Route 1	52-4-42-19-40-41-13-25-18-57	89.298
Route 2	52-17-37-44-15-45-33-39-10-49-5-12-57	102.260
Route 3	55-22-8-26-31-28-3-36-35-20-2-60	93.768
Route 4	55-29-21-16-50-34-30-9-38-11-32-1-60	92.883
Route 5	52-14-24-43-7-23-48-6-27-46-47-57	107.410
Christo-75c-10d		
Routes		**C_r**
Route 1	76-26-67-86	12.902
Route 2	84-38-11-94	16.657
Route 3	80-61-64-42-41-43-1-22-90	86.886
Route 4	80-28-74-30-48-21-90	52.929
Route 5	84-65-66-59-14-35-53-94	64.007
Route 6	76-68-2-62-73-33-6-75-86	57.739
Route 7	76-7-8-19-54-13-57-15-29-45-86	81.903
Route 8	84-58-72-39-9-25-55-31-10-94	93.016
Route 9	76-63-23-56-49-24-18-50-17-86	115.028
Route 10	76-34-46-52-27-4-86	33.575
Route 11	76-51-16-3-44-32-40-12-86	67.872
Route 12	80-69-71-60-70-20-37-5-47-36-90	64.812
Christo-100c-10d		
Routes		**C_r**
Route 1	108-4-118	20.881
Route 2	108-80-68-77-3-79-33-81-9-51-50-76-12-118	71.328
Route 3	108-54-24-29-78-34-35-71-65-66-20-30-70-1-69-27-28-118	138.990
Route 4	108-55-25-39-67-23-56-75-41-22-74-72-73-21-118	96.541
Route 5	108-26-53-58-13-94-95-97-87-42-43-15-57-2-40-118	100.213
Route 6	102-89-6-96-99-59-92-98-37-100-91-85-93-5-60-112	59.716
Route 7	102-8-46-47-36-49-64-11-19-48-82-112	105.173
Route 8	102-52-31-10-32-90-63-62-88-7-112	75.161
Route 9	102-83-84-61-16-44-14-38-86-17-45-18-112	101.467
Perl-12c-2d		
Routes		**C_r**

Continued on next page

Table A.1 – continued from previous page

Route 1	13-9-8-6-1-2-3-7-15	136.858
Route 2	13-4-5-11-12-10-15	118.725
Perl-55c-15d		
Routes		**C_r**
Route 1	57-14-27-54-39-38-16-72	652.563
Route 2	57-12-28-23-17-22-72	441.890
Route 3	63-52-50-53-47-10-11-78	675.342
Route 4	67-25-48-35-24-26-36-82	483.766
Route 5	63-1-13-78	88.463
Route 6	67-18-29-19-31-7-15-82	447.111
Route 7	63-8-3-30-33-32-45-78	433.403
Route 8	63-44-46-40-55-43-34-78	534.471
Route 9	63-5-4-9-6-42-2-78	325.753
Route 10	67-49-51-37-21-20-41-82	662.748
Perl-85c-7d		
Routes		**C_r**
Route 1	87-72-74-73-12-71-70-14-63-94	565.929
Route 2	87-62-61-65-66-68-69-27-16-94	567.447
Route 3	89-5-1-44-46-43-34-3-8-96	530.845
Route 4	87-32-38-2-45-30-33-94	378.150
Route 5	87-22-17-28-75-23-29-19-94	521.711
Route 6	91-49-76-51-79-13-67-83-82-98	795.697
Route 7	89-9-6-15-85-31-7-42-4-96	469.654
Route 8	91-48-35-26-84-18-36-25-41-98	673.634
Route 9	89-47-53-50-81-80-52-64-11-96	605.381
Route 10	91-60-10-77-37-78-21-20-24-98	461.064
Route 11	87-54-59-58-39-56-57-40-55-94	811.106
Min-27c-5d		
Routes		**C_r**
Route 1	29-5-6-17-16-24-22-23-25-4-34	944.601
Route 2	30-3-8-7-2-35	167.015
Route 3	30-10-9-11-26-27-1-35	456.097
Route 4	29-14-20-21-18-19-12-13-15-34	950.304
Min-134c-8d		
Routes		**C_r**
Route 1	137-1-2-28-3-14-121-122-113-130-120-145	421.071
Route 2	137-132-109-119-127-108-128-134-131-129-125-124-145	326.117
Route 3	139-100-99-95-106-94-104-98-96-105-97-83-93-107-103-91-89-82-147	872.775
Route 4	139-102-101-126-42-117-114-133-115-44-36-34-38-40-30-31-87-45-85-147	630.991
Route 5	136-64-63-29-46-54-48-62-65-51-52-7-13-16-24-19-22-144	492.026
Route 6	136-26-37-32-144	47.658

Continued on next page

Table A.1 – continued from previous page

Route 7	136-35-43-33-144	234.2798
Route 8	138-70-78-72-77-76-73-80-75-92-71-79-69-90-81-86-146	718.366
Route 9	137-118-111-110-112-123-116-145	170.5329
Route 10	138-84-88-50-59-66-55-58-60-61-74-146	381.7935
Route 11	136-39-57-47-49-68-67-56-53-144	548.5576
Route 12	136-17-20-12-27-5-18-25-10-11-9-4-6-21-8-15-23-144	193.2271
Route 13	139-41-147	178.0449
Daskin-88c-8d		
Routes		**C_r**
Route 1	95,74,2,32,6,46,11,14,39,41,78,68,30,81,21,65,85,87,103	100.998
Route 2	93,75,64,17,58,3,34,69,18,60,4,52,24,55,59,66,15,45,83,101	68.711
Route 3	93,42,51,56,47,57,77,26,76,62,9,33,38,22,10,27,8,28,29,43,50,67,31,80,84,101	107.335
Route 4	95,1,63,20,61,86,79,88,70,49,103	28.893
Route 5	93,5,37,53,71,40,101	20.236
Route 6	93,72,12,82,19,54,101	6.064
Route 7	95,7,23,16,73,35,36,25,44,13,48,103	23.843
Daskin-150c-10d		
Routes		**C_r**
Route 1	159,2,169	1638.508
Route 2	159,142,169	206.475
Route 3	152,94,57,58,4,14,25,64,92,127,67,52,21,146,162	4943.885
Route 4	152,10,135,34,112,7,113,121,83,123,11,97,42,95,49,162	4586.064
Route 5	152,101,12,41,20,133,118,108,6,122,141,162	2692.012
Route 6	152,50,99,138,26,89,144,74,82,85,129,66,18,27,9,33,109,162	2361.339
Route 7	159,100,37,71,24,150,61,117,36,103,77,91,132,86,69,3,136,169	7409.612
Route 8	159,45,110,53,120,62,119,143,29,78,60,134,22,63,126,8,105,19,65,169	3660.271
Route 9	159,104,72,16,128,59,35,88,106,39,116,30,81,55,32,90,68,38,40,102,139,169	2191.847
Route 10	151,75,15,137,93,87,54,43,13,148,96,28,56,147,149,73,48,76,47,111,84,70,161	2689.599
Route 11	151,115,44,31,23,107,130,140,46,17,131,5,145,80,125,98,79,51,161	2552.736
Route 12	151,1,114,124,161	301.813
Or-117c-14d		
Routes		**C_r**
Route 1	119-3-133	0.000
Route 2	118-94-9-30-43-98-62-46-92-34-85-21-31-97-57-51-12-4-59-35-47-66-15-58-70-75-29-132	2831.217
Route 3	118-22-53-76-13-54-48-95-17-42-36-11-26-77-14-90-99-64-88-2-52-44-81-132	1355.719
Route 4	119-39-37-41-40-73-72-68-80-63-61-25-74-82-89-116-117-23-102-111-100-103-104-110-109-101-114-115-112-106-108-113-107-105-71-20-27-79-10-56-24-8-49-50-133	6895.812
Route 5	119-1-93-33-67-6-32-16-19-83-38-5-28-45-65-87-86-55-69-7-96-60-78-84-133	801.994
Route 6	122-18-91-136	12.649
Srivastava-8c-2d		
Routes		**C_r**

Continued on next page

Table A.1 – continued from previous page

Route 1	9-6-1-8-4-11	223.214
Route 2	10-3-7-2-5-12	180.451

VITA

SUAT BOĞ was born in Samsun, Turkey on September 5, 1981. He graduated from Samsun Anatolian High School in 1999. He received his B.Sc. degree in Industrial Engineering from Boğaziçi University, Istanbul, in July 2004. From September 2004 to July 2006, he worked as a teaching and research assistant in Koç University, Turkey.