ABSTRACT

Title of dissertation: A STUDY OF FOUR NETWORK PROBLEMS IN TRANSPORTATION, TELECOMMUNICATIONS, AND SUPPLY CHAIN MANAGEMENT

Si Chen, Doctor of Philosophy, 2007

Dissertation directed by: Professor Bruce Golden and
Associate Professor Subramanian Raghavan
Robert H. Smith School of Business

The increasing material costs and the rapid advances in computing technology have both motivated and promoted the study of network problems that arise in several different application domains. This dissertation consists of four chapters on network applications in transportation, telecommunications, and supply chain management. The core of our research is to apply heuristic search procedures and combinatorial optimization techniques to various practical problems.

In the second chapter we investigate the split delivery vehicle routing problem (SDVRP), where a customer's demand can be split among several vehicles. The third chapter deals with the regenerator location problem (RLP) that arises in optical networks. The fourth chapter solves the parametric uncapacitated network design problems on series-parallel graphs, which have potential application in supply chain management. In the fifth chapter we study the arc routing problem that arises in the small package delivery industry. The last chapter summarizes the dissertation.

The results in this dissertation indicate that the methodologies developed to

solve the network problems in the four different applications are quite efficient. Consequently, when applied in practice, they have the potential to significantly improve the operational efficiency of organizations in the relevant application domains.

# A STUDY OF FOUR NETWORK PROBLEMS IN TRANSPORTATION, TELECOMMUNICATIONS, AND SUPPLY CHAIN MANAGEMENT

by

Si Chen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:
Professor Bruce Golden, Co-Chair
Professor Subramanian Raghavan, Co-Chair
Professor Jeffrey Herrmann
Professor Edward Wasil
Dr. Richard Wong

Dedicated to my dearest Mom and Dad.

# ACKNOWLEDGMENTS

Five years I have spent in the PhD program at the University of Maryland. Many times I dreamed of graduating and starting a new chapter of my life. But now, as the very day I have been so eagerly waiting for finally draws close, I find myself wishing time could slow down a little bit. It is true that we do not know what we have got until we are about to lose it. I owe my happiness, growth and achievements to all those beautiful people around me.

I am forever indebted to my advisors, Dr. Bruce Golden and Dr. S. Raghavan. Much of the dissertation research would have been impossible without their guidance, patience and encouragement.

I owe my deepest gratitude to Dr. Edward Wasil, Dr. Rich Wong and Dr. Hongsheng Zhong, who have provided invaluable knowledge and advice whenever I needed it.

I would also like to thank my dissertation committee member, Dr. Jeffrey Herrmann, for his useful comments.

My special thanks go to my friends, Huiju Zhang, Junjie Liu, Patricia Bowgren, and Victoria Tran, who have helped me grow mentally and emotionally.

Last but not the least, I would like to thank my parents for their unconditional love and support.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Network problems pervade our society. They arise in transportation (the daily commute, trash collection, newspaper delivery), communication (the telephone, email, teleconference), and social networks (the interaction among knowledge workers, control of of infectious diseases, disaster evacuation). The optimization techniques used to analyze and solve these problems are embedded in the technologies and services that make our experience of using telephone, surfing on the Internet and shipping holiday gifts more efficient and pleasant. Indeed, networks provide a rich framework for studying a wide range of practical problems that have an underlying physical or logical network structure.

In the past decade or so, the advances in the research of network problems have had a significant impact on the US economy. For instance, United Parcel Service (UPS), the world's largest package delivery company, saved $276 million over a decade after it redesigned its overnight delivery network using advanced network optimization techniques [64].

The practice of using network models to improve efficiency has an increasingly important role in today's service driven society. The reason is twofold. First, the rising material costs, as represented by the recent spike in gasoline and diesel price,

are putting a tremendous burden on the global economy. Furthermore, in developed countries such as the United States, organizations also face high labor costs. As a result, higher efficiencies via cost reduction has become a necessity. Second, the advances in computing technology, as demonstrated by the state-of-the-art Blue Gene system which has a peak speed of 360 Teraflops - 360 trillion calculations a second [5], enable researchers and practitioners to tackle much larger and more complicated problems. In summary, the growing pressure for cost reduction and the continuous advances in computing technology have provided great incentives and numerous opportunities for the studies and applications of network optimization.

However, though conceptually simple — one may ask how to find the minimal-mileage route to deliver all the mail in College Park — these network problems are often combinatorial in nature. Instances of these problems are believed to be hard to solve optimally in general. Therefore, in practice, we often rely on efficient heuristics to provide high quality solutions.

In this dissertation we address four network problems that arise in the transportation, telecommunications, and supply chain management application domains. We develop heuristic search procedures and combinatorial optimization techniques to solve these problems.

## 1.2 Outline Of The Dissertation

In Chapter 2, we investigate the split delivery vehicle routing problem (SDVRP), where a customer's demand can be split among several vehicles. We review

applications of the SDVRP including the routing of helicopters in the North Sea and solution methods such as integer programming and tabu search. We develop a new heuristic that combines a mixed integer program and a record-to-record travel algorithm. Our heuristic produces high-quality solutions to six benchmark problems that have 50 to 199 customers and it generally performs much better than tabu search. On five other problems for which lower bounds exist, our heuristic obtains solutions within 5.85%, on average. Finally, we generate 21 new test problems that have 8 to 288 customers. A near-optimal solution can be visually estimated for each problem. We apply our heuristic to these new problems and report our computational results.

In Chapter 3, we deal with the regenerator location problem (RLP) that arises in optical networks. The ever-increasing usage of digital communications has inspired the development of a variety of new applications in business and consumer markets. These new applications, which usually involve voice and multimedia services, add a significant amount of traffic to telecommunications networks. As a result providers have built optical networks that have significant capacity and flexibility to meet the demands of the network users. In an optical network, the distance a signal may be sent from a source to a destination is limited by transmission impairments (the signal strength deteriorates as it gets further from the source). Let $d_{max}$ denote the maximum distance a signal can travel before its quality deteriorates. In order to deal with this situation, whenever a signal has to travel further than $d_{max}$ it is amplified by the use of regenerators. Regenerators may be installed only at nodes of the network. As the cost of regenerators is very high we wish to deploy as

few regenerators as possible, while ensuring all nodes can communicate with each other (i.e., send a signal to each other). We prove that RLP is NP-complete and develop three heuristics that produce high-quality solutions for large-sized networks in a matter of seconds. We also study a generalization of RLP called the generalized regenerator location problem (GRLP). In RLP all nodes must be connected and all nodes are potential regenerator locations. In GRLP, a set of terminal nodes T must be connected and the potential regenerator locations are the set of nodes S. Here, $S \cup T = N$, $S \cap T \neq N$. We propose two efficient heuristics for GRLP.

In Chapter 4, we solve the parametric uncapacitated network design problems on series-parallel graphs. In many business applications, competitive imperatives are driving decision-makers to consider multiple opposing criteria or objectives. For instance, they often need to consider the trade-offs between investment and operational cost. Under such circumstances, the goal is to find an efficient, or Pareto optimal solution, which satisfies the opposing objectives. In this chapter, we consider the parametric uncapacitated network flow problem on a series-parallel graph and propose a polynomial time algorithm. This network design problem has potential applications in supply chain management. For instance, the well known economic lot-sizing problem may be modeled as a network flow problem on a series-parallel graph. It also generalizes earlier work on a bicriteria product design problem (see Raghavan et al. (2002)) which may be modeled as a parametric shortest path problem on a series-parallel graph. Our results build upon Wald (1999) and Raghavan et al. (2002).

In Chapter 5, we study the arc routing problem that arises in the small package

delivery industry. In practice, the service providers are encouraged to follow a master route, which defines a sequence of street segments and the direction in which each street segment is to be traversed for his/her delivery area. The benefit of using master route is two-fold: (i) it maintains the consistency of a route in the sense that the driver follows similar route each day, and that regular customers are serviced by the driver and at approximately the same time each day. (ii) It improves the efficiency of delivery since packages are loaded into the vehicles in accordance with the master route. Our overall objective is to construct efficient master routes for the service providers. Given an extended period of time (more than one day), if the set of street segments requesting services every day remains unchanged, we only need to solve an arc routing problem once during the entire time horizon. However, in reality, the street segments that need to be visited vary on a daily basis. Currently, this problem is often approached in a deterministic manner and the resulting master route is used over the entire planning horizon. However, the uncertainty as to whether a street segment requires service on a particular day suggests that it may be beneficial to study the problem in a probabilistic context. We describe two new approaches to model this problem and present the computational results from case studies on problems derived from real-world service routes as well as computer generated problems.

In Chapter 6, we summarize the contributions of this dissertation.

# Chapter 2

# Split Delivery Vehicle Routing Problem

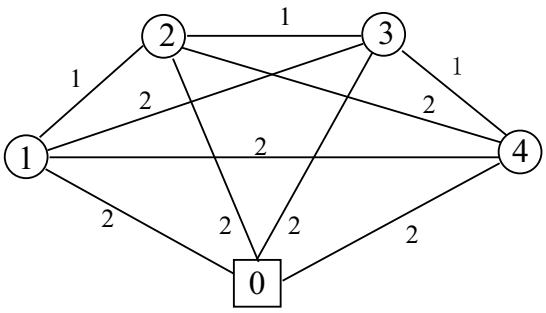
## 2.1 Introduction

In the standard version of the vehicle routing problem (VRP), a homogeneous fleet of vehicles is based at a single depot. Each vehicle has a fixed capacity and must leave from and return to the depot. Each customer has a known demand and is serviced by exactly one visit of a single vehicle. A sequence of deliveries (known as a route) must be generated for each vehicle so that all customers are serviced and the total distance traveled by the fleet is minimized. Recent algorithmic developments and computational results for heuristics that solve the VRP are covered by Cordeau et al. (2005).

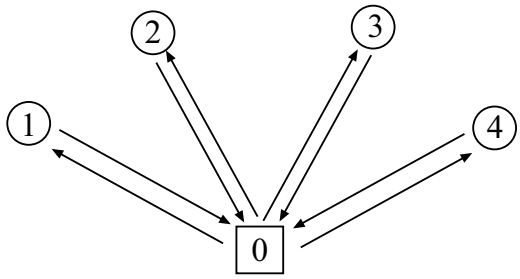
In the split delivery vehicle routing problem (SDVRP), a customer can be serviced by more than one vehicle, that is, a customer's demand can be split among several vehicles. By allowing split deliveries, the potential exists to use fewer vehicles and to reduce the total distance traveled by the fleet. We point out that, in general, for both the VRP and SDVRP, using fewer vehicles does not necessarily reduce the total distance (even if the triangle inequality holds). The SDVRP is NP-hard (Dror and Trudeau 1990) and is difficult to solve optimally. In Figure 2.1, we show an example from Archetti, Hertz, and Speranza (2006) that illustrates the savings in distance and number of vehicles that can be achieved by using split deliveries.In this

chapter, we consider an undirected graph (our algorithm also works for a directed graph). Over the last decade or so, researchers have modeled practical problems as SDVRPs. Mullaseril, Dror, and Leung (1997) study the distribution of feed to cattle at a large livestock ranch in Arizona. About 100,000 head of cattle are kept in large pens that are connected by a road network. Six trucks deliver feed to the pens within a specified time window each day. Due to feed weighing and loading inaccuracies, the last pen on a route may not receive its full load and would need to have the rest of its load delivered by a second truck on a different route. The authors model this feed distribution problem as a capacitated rural postman problem with split deliveries and time windows. They develop a solution algorithm that uses k-split interchange and route addition (more about these in the next section). The results of computational experiments with five types of feed and time windows show that allowing split deliveries significantly reduces the total distance travelled by the fleet in four of five cases.

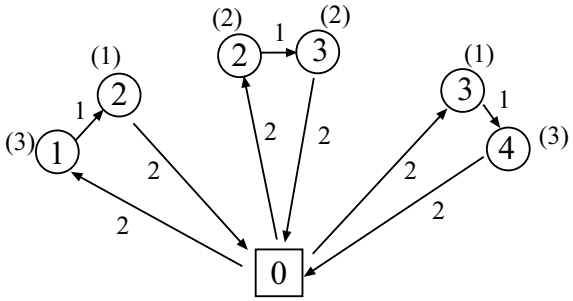
In the North Sea, off the coast of the Netherlands, there are 51 platforms for the production of natural gas. Each platform has a regular crew of 20 to 60 employees and each employee works every other week. Crew members are exchanged each week by helicopters based at an airport near Amsterdam. One person leaving the platform is exchanged for one person arriving for work at the platform. Each helicopter has a fixed capacity of 27 seats (23 seats are available for crew exchanges and 4 seats are left free for emergencies and cargo) and there are enough helicopters to make all of the required exchanges in a week. In addition, each helicopter has a specified fuel capacity that limits the range it can fly.




(a) Note 0 is the depot. Customers 1,2,3,4 have a demand of 3 units.
Vehicle capacity is 4 units. Distance between i and j is adjacent to the edge.




(b) VRP optimal solution with four vehicles and a total distance of 16




(c) SDVRP optimal solution with three vehicles and a total distance of 15

Figure 2.1: *Example from Archetti, Hertz, and Speranza (2006) showing split deliveries.*

Sierksma and Tijssen (1998) model crew exchanges as a discrete split delivery routing problem. They formulate the problem as an integer program and solve a relaxed linear program using column generation and a rounding procedure to produce integer solutions (denoted by CGRP). In addition, they develop a cluster-and-route procedure (denoted by CR).

Sierksma and Tijssen conduct a computational experiment with 11 different simulated data sets (51 platforms and 2 to 50 crew exchanges per platform). CGRP, CR, a sweep heuristic, and the Clarke and Wright algorithm are applied to each problem and the resulting solution is then improved by several post-processors including two-opt. Over all 11 instances, CGRP has the smallest average deviation from a lower bound for each problem, closely followed by CR. However, the running time of CR is very fast (a few seconds for the entire procedure), while the running time of CGRP is very slow (about 50 seconds for one solution).

Song, Lee, and Kim (2002) consider the distribution of newspapers from printing plants to agents. They carry out a case study for Chosun Ilbo which is a major newspaper in South Korea. Chosun Ilbo has three printing plants and 400 agents around Seoul, and it prints three local editions. Printing starts around midnight and finishes by 3:00 am. Newspapers are then dispatched to agents at 2:30 am, 3:00 am, and 3:30 am. Agents insert supplements into the newspapers, which are then delivered to residential (home) customers.

Some agents who are close to the printing plants are considered for split delivery of newspapers with a first dispatch at 2:30 am and a second dispatch at 3:30 (at most two deliveries are allowed). The split delivery spreads out the work of insert-

ing supplements and lets home delivery start earlier. Agents far from the printing plants and agents with small demands are not considered for split deliveries.

Song, Lee, and Kim use a two-phase solution procedure. In Phase I, they allocate agents to plants by solving a 0-1 integer programming problem. In Phase II, the authors determine the split deliveries, generate the vehicle routes using a modified savings rule and a weighted savings rule, and schedule the vehicles for dispatch. In the case study for Chosun Ilbo, the authors use a geographic information system to obtain road distances. When compared to the manual method used by Chosun Ilbo, the allocation, routing, and scheduling procedure reduces the delivery cost by an average of 15%.

In commercial sanitation collection, businesses and organizations often place their trash in large containers or bins. An office building may have several of these bins. Each bin is lifted and its contents are emptied into the trash truck. Since the bins are large, several trucks may be required to pick up all of the trash at a particular office building. However, a bin's load cannot be split. This problem can be modeled in two ways. Either each bin is considered as a separate demand or the office building is handled as a single demand. In the latter case, a discrete number of splitting options is allowed (Levy 2006).

In Sec. 2.2, we review algorithms that have been used recently to solve the SDVRP. In Sec. 2.3, we formulate a mixed integer program for the SDVRP. In Sec. 2.4, we develop a new heuristic for solving the SDVRP that combines our mixed integer program and a record-to-record travel algorithm. We report computational results for our new heuristic on benchmark problems that have 50 to 199 customers

and on five other problems for which lower bounds exist. In Sec. 2.5, we generate 21 new test problems that have 8 to 288 customers. A near-optimal solution can be visually estimated for each problem. We apply our new heuristic to these 21 problems and report our results. In Sec. 2.6, we summarize our contributions.

## 2.2 Solution Approach

Over the last 15 years or so, numerous procedures including tabu search have been used to solve the SDVRP. In this section, we review the procedures and their computational results.

Dror and Trudeau (1989, 1990) develop an algorithm for solving the SDVRP that uses a k-split interchange and route addition. A k-split interchange splits the demand of customer i among k routes as long as the sum of the spare (or residual) capacities of the k routes is not less than the demand of customer i and there is a positive savings in distance when customer i is removed from its current route and added to the k routes. Route addition serves as an inverse operation to a k-split interchange. Add a route to eliminate the split delivery of a customer if the addition of the route reduces the total distance.

The authors use a two-stage algorithm to solve the SDVRP. In Stage 1, an initial solution to the VRP is constructed and then customer interchanges and route improvements are considered. In Stage 2, k-split interchanges, route additions, customer interchanges, and route improvements are used to produce a solution to the SDVRP.

Dror and Trudeau conduct computational experiments on three problems with 75, 115, and 150 customers and a vehicle capacity of 160. The 75-point problem is taken from Eilon, Watson-Gandy, and Christofides (1971). The 150-point problem and 115-point problem are obtained from the 75-point problem. Customer demand is generated according to six scenarios: [0.01 - 0.1], [0.1 - 0.3], [0.1 - 0.5], [0.1 - 0.9], [0.3 - 0.7], [0.7 - 0.9]. For example, in the first scenario (very small customer demand), the demand for customer i is randomly selected from a uniform distribution on the interval [160(0.01), 160(0.1)] = [1.6, 16]. For each problem (75, 115, 150 customers), 30 instances are generated for each of the six scenarios. The two-stage algorithm is then applied to the 180 instances for each size problem. Dror and Trudeau solve each problem twice - as an SDVRP and a VRP. They observe that, when customer demand is small relative to vehicle capacity, there are almost no split deliveries. When customer demand is very large (e.g., [0.7 - 0.9]), split deliveries occur thereby reducing the total distance traveled (e.g., for the 75-customer problem, an average distance savings of 11.24% over the VRP solution) and reducing the number of vehicles used (e.g., for the 75-customer problem, an average reduction of 14.07 vehicles over the VRP solution).

Frizzell and Giffin (1992) consider the SDVRP on a grid network and introduce upper and lower bounds on the size of a split delivery to a customer. They develop a construction heuristic and a splitting cost heuristic that assigns specific splitting costs for each customer. Frizzell and Giffin test their heuristics on a set of 1050 problems and find that the solutions generated by their construction heuristic reduce the total distance and number of vehicles when compared to the solutions of a greedy

construction heuristic.

Dror, Laporte, and Trudeau (1994) formulate the SDVRP as an integer linear program with several new classes of valid constraints. The authors develop a constraint relaxation algorithm using branch and bound for solving the SDVRP exactly that uses the solution from the algorithm of Dror and Trudeau (1989) as a first upper bound on the optimal solution value of the SDVRP. Computational experiments are run on three problems with 10, 15, and 20 customers and varying customer demands taken from the 75-point problem of Eilon, Watson-Gandy, and Christofides (1971). The computational results show that various constraints could successfully reduce the gap between the lower and upper bounds of the optimal solution value of SDVRP. In addition, the results establish the high quality of the solutions generated by the Dror and Trudeau algorithm.

Frizzell and Giffin (1995) study the SDVRP with time windows where customers are located on a grid network. They develop a construction heuristic for solving the problem. The solution that emerges from the construction heuristic is improved by either removing one customer from a route and inserting it on another route or exchanging two customers from two routes. The authors generate 6,480 problems and use five measures including drive time and number of routes to evaluate the performance of their heuristic. They also report results on six benchmark VRPs that have time windows.

Belenguer, Martinez, and Mota (2000) propose a lower bound for the SDVRP based on a polyhedral study of the problem. They introduce a new family of valid inequalities and propose a cutting-plane algorithm for generating a lower bound to

the SDVRP. The algorithm is tested on 11 instances from TSPLIB and 14 random instances with 50, 75, and 100 customers (these instances are similar to the problems in Dror and Trudeau (1989)). The algorithm performs reasonably well with the average gap between the upper bound and the lower bound for the TSPLIB problems about 3% and about 8% for the random instances.

Archetti, Hertz, and Speranza (2006) develop a tabu search algorithm (called SPLITABU) for solving the SDVRP. Their algorithm has three phases: (1) Initial solution phase. Create as many direct trips to customers as possible and then solve a giant TSP tour using the GENIUS algorithm to produce a feasible solution. (2) Tabu search phase. Remove a customer from a current set of routes and insert it on a new route or an existing route with available capacity in the cheapest way. Consider inserting a customer on a route without removing it from its current route. (3) Final improvement phase. Improve the solution from the second phase (e.g., apply the GENIUS algorithm to individual routes). In a variant called SPLITABU-DT, the authors improve solutions using the customer (node) interchanges of Dror and Trudeau (1989, 1990) and two-opt. There are only two parameters that need to be set in SPLITABU: the length of the tabu list and the maximum number of iterations.

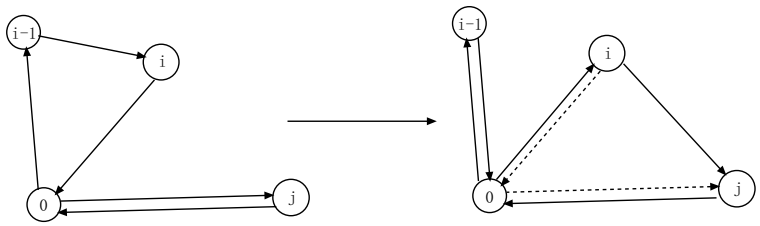
Archetti, Hertz, and Speranza test three variants of SPLITABU on seven problems with 50 to 199 customers (the problems are obtained from seven classical capacitated VRP instances - instances 1, 2, and 3 from Christofides and Eilon (1969) and instances 4, 5, 11, and 12 from Christofides, Mingozzi, and Toth (1979) - by considering the original demands of the customers and demands defined according to the rules proposed by Dror and Trudeau (1989, 1990)), run each variant five times

on each problem, and compare the results produced by the variants to the results produced by Dror and Trudeau's algorithm (denoted by DT). Overall, SPLITABU-DT is the clear winner with a smaller total distance traveled than DT for every problem. On average, the best solutions produced by SPLITABU-DT are 5.38% lower than the solutions generated by DT.

We point out that the recent dissertations by Liu (2005) and Nowak (2005) model variants of the SDVRP, develop solution procedures, and report computational results.

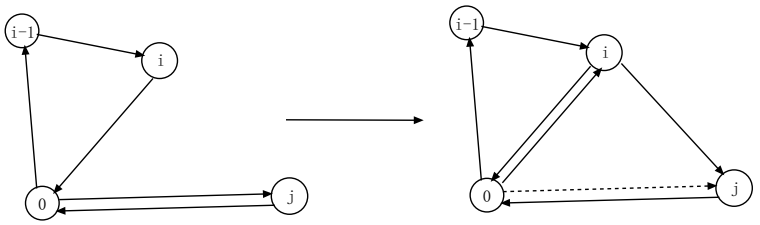
### 2.2.1 An Endpoint Mixed Integer Program (MIP) Formulation

We start with an initial solution to the SDVRP, say a solution from the Clarke and Wright (1964) algorithm to the corresponding VRP instance. For each route in the initial solution, we consider its one or two endpoints and the c closest neighbors to each endpoint (c is a parameter whose value we need to set and the neighbors are endpoints). Each endpoint is allowed to reallocate its demand among its neighbors. After reallocation, there are three possibilities for each endpoint (we assume symmetric distances here): (1) No change is made. (2) The endpoint $i$ is removed from its current route(s) and all of its demand is moved to another route or routes. (3) The endpoint $i$ is partially removed from its current route(s) and part of its demand is moved to another route or routes. The reallocation process and the savings that result are illustrated in Figure 2.2. We formulate an endpoint mixed integer program (EMIP) whose objective is to find a set of reallocation operations




savings = $l_{(i-1),\,i} + l_{i,\,0} - l_{(i-1),\,0}$

(a)endpoint i removed from its current route and
all of its demand is moved to another route




savings = - ($l_{0,\,i} + l_{i,\,j} - l_{0,\,j}$)

(a)endpoint i removed from its current route and
all of its demand is moved to another route

Figure 2.2: *Reallocating the demand of endpoint $i$ to other routes.*

that maximizes the total savings. Let $i$ and $j$ be the endpoints of current routes, $l_{ij}$ is the distance between $i$ and $j$ (we use Euclidean distances in our computational experiments; other types of distances could be used), $R_i$ is the residual capacity on the route with $i$ as an endpoint, $D_i$ is the demand of endpoint $i$ carried on its route, $p(i)$ is the predecessor of endpoint $i$, $s(i)$ is the successor of endpoint $i$, $R$ is the set of routes, $N$ is the set of endpoints, and $\mathrm{N}C(i)$ is the set of $c$ closest neighbors of endpoint $i$.

The decision variables are defined next. Let $d_{ij}$ be the amount of $i$'s demand moved before endpoint $j$, $m_{ij}$ equals 1 if endpoint $i$ is inserted before endpoint $j$ and equals 0 otherwise, and $b_i$ equals 1 if endpoint $i$'s entire demand is removed from the route on which it was an endpoint and equals 0 otherwise.

The objective function and constraints of the EMIP are given by the following.

$$\max \sum_{i \in N} b_i(l_{is(i)} + l_{p(i)i} - l_{p(i)s(i)}) - \sum_{i \in N} \sum_{j \in NC(i)} m_{ij}(l_{ij} + l_{p(j)i} - l_{p(j)j}) \tag{2.1}$$

subject to

$$\sum_{i:j \in NC(i)} d_{ij} + \sum_{q:k \in NC(q)} d_{qk} - \sum_{l \in NC(k)} d_{kl} - \sum_{t \in NC(j)} d_{jt} \leq R_r \; \forall r \in R \tag{2.2}$$

$k$, $j$ are two endpoints of route $r$

$$\sum_{j \in NC(i)} d_{ij} \leq D_i \; \forall i \in N \tag{2.3}$$

$$\sum_{j \in NC(i)} d_{ij} \geq D_i \times bi \; \forall i \in N \tag{2.4}$$

$$D_i m_{ij} \geq d_{ij} \; \forall i \in N, j \in NC(i) \tag{2.5}$$

$$1 - b_i \geq \sum_{j:i \in NC(j)} m_{ji} \; \forall i \in N \tag{2.6}$$

$$1 - b_{p(i)} \geq \sum_{j:i \in NC(j)} m_{ji} \; \forall i \in N \tag{2.7}$$

$$b_k + b_j \leq 1 \; \forall r \in R; \tag{2.8}$$

$k$, $j$ are two endpoints of route $r$

$$d_{ij} \geq 0 \; \forall i \in N, j \in NC(i) \tag{2.9}$$

$$m_{ij} = 0, 1 \; \forall i \in N, j \in NC(i) \tag{2.10}$$

$$b_i = 0, 1 \; \forall i \in N \tag{2.11}$$

The objective function (2.1) maximizes the total savings from the reallocation process. In (2.2), the amount added to a route minus the amount taken away from a route must be less than or equal to the residual capacity. In (2.3), the amount

diverted from an endpoint on a route must be less than or equal to the demand of that endpoint on the route. In (2.4), if an endpoint is removed from a route, then all of its demand must be diverted to other routes. In (2.5), if $d_{ij} > 0$ (that is, we move some of node $i$'s demand before node $j$), then $m_{ij} = 1$ (that is, node $i$ is inserted before node $j$). In ((2.6), if node $i$ is removed from a route, then no node can be inserted before node $i$. In (2.7), if the predecessor of node $i$ is removed from a route, then no node can be inserted before node $i$. In addition, from (2.6) and (2.7), if endpoints $i$ and $p(i)$ are not removed from their route, then at most one endpoint can be inserted before endpoint $i$. In (2.8), if a route has only two endpoints, then we cannot remove both endpoints at the same time.

We now present the EMIP formulation of the small example given in Figure 2.3(a).

$$\max\ 2b_1l_{01} + 2b_2l_{02} + 2b_3l_{03} - m_{12}(l_{01} + l_{12} - l_{02}) - m_{13}(l_{01} + l_{13} - l_{03})$$

$$-m_{21}(l_{02} + l_{21} - l_{01}) - m_{23}(l_{02} + l_{23} - l_{03}) - m_{31}(l_{03} + l_{31} - l_{01})$$

$$-m_{32}(l_{03} + l_{32} - l_{02})$$

subject to

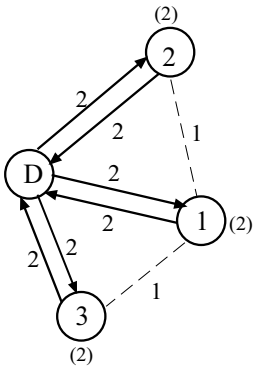
$$d_{21} + d_{31} - d_{12} - d_{13} \leq R_1$$

$$d_{12} + d_{32} - d_{21} - d_{23} \leq R_2$$

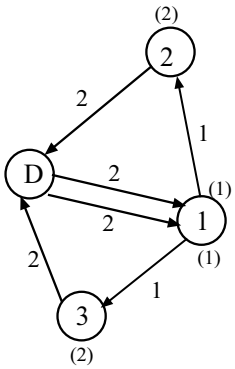
$$d_{13} + d_{23} - d_{31} - d_{32} \leq R_3$$

$$d_{12} + d_{13} \leq D_1$$

$$d_{21} + d_{23} \leq D_2$$




(a) node 0 is the depot. Customer 1, 2 and 3 have a demand of two units each. Vehicle capacity is three units. Distance between nodes is adjacent to the edge. The Clark-and-Wright solution is given by the solid edges and has a total distance of 12.




(b) Optimal solution. The demand of customer 1 is split between two routes. The total distance is 10.

Figure 2.3: *Small example to illustrate the EMIP formulation and an optimal solution.*

$$d_{31} + d_{32} \leq D_3$$

$$d_{12} + d_{13} \geq D_1 b_1$$

$$d_{21} + d_{23} \geq D_2 b_2$$

$$d_{31} + d_{32} \geq D_3 b_3$$

$$D_1 m_{12} \geq d_{12}$$

$$D_1 m_{13} \geq d_{13}$$

$$D_2 m_{21} \geq d_{21}$$

$$D_2 m_{23} \geq d_{23}$$

$$D_3 m_{31} \geq d_{31}$$

$$D_3 m_{32} \geq d_{32}$$

$$1 - b_1 \geq m_{21} + m_{31}$$

$$1 - b_2 \geq m_{32} + m_{12}$$

$$1 - b_3 \geq m_{23} + m_{13}$$

$$d_{ij} \geq 0$$

for $i$, $j = 1, 2, 3$

$$b_i,\, m_{ij} = 0, 1$$

for $i$, $j = 1, 2, 3$

For this example, we have $R_i = 1$ and $D_i = 2$ for $i = 1, 2, 3$. The distances are given by $l_{01} = 2$, $l_{02} = 2$, $l_{03} = 2$, $l_{10} = 2$, $l_{12} = 1$, $l_{13} = 1$, $l_{20} = 2$, $l_{21} = 1$, $l_{23}$

$= 2$, $l_{30} = 2$, $l_{31} = 1$, and $l_{32} = 2$ and the objective function is

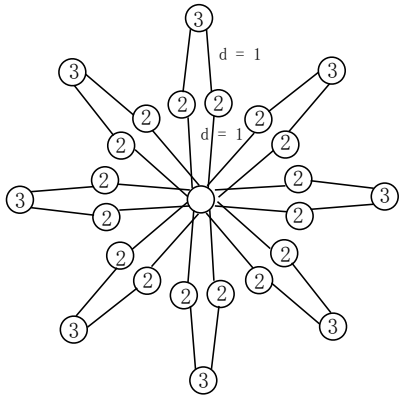
$$\max\ 4b_1 + 4b_2 + 4b_3 - m_{12} - m_{13} - m_{21} - m_{23} - m_{31} - m_{32}$$

Since all of the routes in the initial Clarke-and-Wright solution have only one endpoint, the constraints in (2.7) and (2.8) are not needed in this example. An optimal solution is given in Figure 2.3(b) and it has a total distance of 10 (there are multiple optimal solutions for this problem).

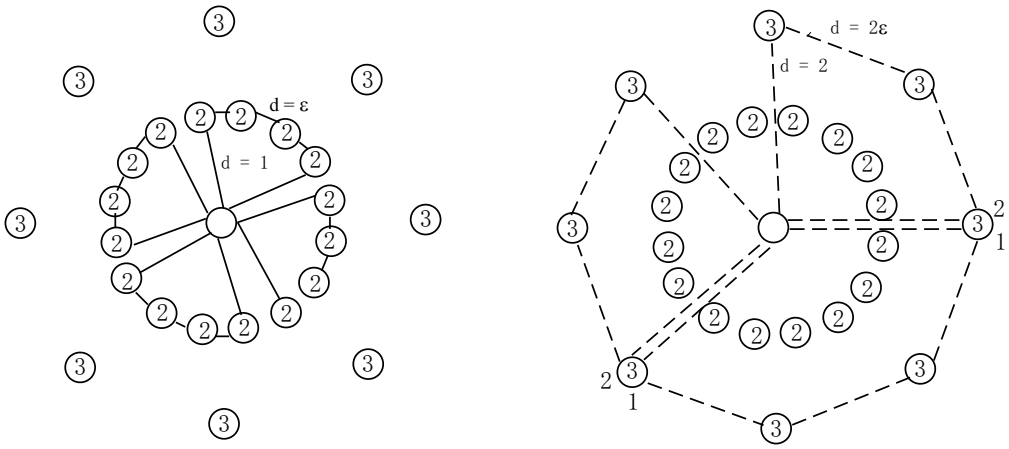
We point out that, for some problems, not all feasible solutions can be reached by solving the endpoint mixed integer program. We illustrate this limitation of the EMIP with a non-Euclidean problem in Figure 4. In Figure 2.4(a), an initial solution uses eight vehicles and has a total distance of 32. In Figure 2.4(b), an improved solution uses seven vehicles, splits two demands, and has a total distance of $20 + 26\epsilon$ which is less than 32 when $\epsilon < \frac{6}{13}$. This improved solution cannot be generated by the EMIP because the customers with a demand of three units are not endpoints and, therefore, cannot be split.

### 2.2.2 Proposed Heuristic

Our new heuristic uses the classical Clark-and-Wright (CW) algorithm to generate a starting solution (it is discussed in greater detail in Appendix A). Using the CW solution, an EMIP is formulated and solved. In this first EMIP, the size of the neighbor list depends on the number of endpoints in the starting solution (e.g., when there are 24 to 120 endpoints, the size of the neighbor list is 10). Furthermore, the endpoint mixed integer programs can be considerable in size and difficult to solve.




(a) Initial solution uses eight vehicles. Each vehicle has a capacity of eight. The demand of each customer (written inside the circles representing nodes) is either two units or three units. With a distance of 1 for each edge (d=1), the total distance of this solution is 32.




(b) Improved solution uses seven vehicles, splits two demands, and has a total distance of 20 + 26ε. When ε < 6/13, the total distance is less than 32.

Figure 2.4: *Example to illustrate the limitation of EMIP.*

Based on our computational work, a 200-node problem, with a vehicle capacity of 200 units and a demand between 140 and 180 units for each customer, has an average of 200 endpoints, 2,200 integer variables, 2,000 continuous variables, and 2,800 constraints in the EMIP. We set a limit of T seconds for solving the first EMIP and save the best feasible solution found during a run. We denote this solution by E1 and point out that it may not be the optimal solution.

Using E1 as the initial solution, a second EMIP is formulated and solved (we denote the solution by E2) with a larger size for the neighbor list and a smaller limit for the running time than those for the first EMIP. The final solution is obtained by post-processing E2 with a variable length record-to-record travel algorithm (VRTR) that was developed by Li, Golden, and Wasil (2005). For each customer i, all edges (for customers in the neighbor list) with length greater than 0.8 L are removed, where L is the maximum length among all edges in i's neighbor list. VRTR considers one-point, two-point, and two-opt moves within and between routes. The details of our new heuristic (denoted by EMIP+VRTR) are given in Figure 2.5.

## 2.3 Computational Results

### 2.3.1 Six Benchmark Problems

We select six problems (1, 2, 4, 5, 11, 12) from Christofides and Eilon (1969) and Christofides, Mingozzi, and Toth (1979) with 50, 75, 100, 120, 150, and 199 customers. For each problem, we generate a customer's demand according to the six scenarios ([0.01 - 0.1], [0.1 - 0.3], [0.1 - 0.5], [0.1 - 0.9], [0.3 - 0.7], [0.7 - 0.9])

**Step 0. Initialization.**
Parameters are I, K, NL (size of the neighbor list), and T (time limit in seconds).
Set $I = 30$ and $K = 5$.
**Step 1. Starting solution.**
Generate a feasible solution using the Clarke-and-Wright algorithm.
N is the number of endpoints in the solution.
If $N < 24$, $NL = N - 1$. If $24 \leq N < 120$, $NL = 10$. If $120 \leq N < 300$, $NL = 8$.
If $N < 24$, $T = \infty$ . If $24 \leq N < 120$, $T = 400$. If $120 \leq N < 300$, $T = 5,000$.
Using the Clarke-and-Wright solution, formulate and solve an EMIP using NL.
If no optimal solution is found after running the EMIP for $T$ seconds, then terminate the EMIP.
Record the best feasible solution found and denote it by E1.
Using E1, formulate and solve an EMIP using $1.5NL$.
If no optimal solution is found after running the EMIP for $0.6T$ seconds, then
terminate the EMIP.
Denote the current solution by E2.
Set Record = objective function value of the current solution.
Set Deviation = 0.01Record.
**Step 2. Improve the current solution.**
For i = 1 to I (I loop)
Do One-Point Move with record-to-record travel, Two-Point Move
with record-to-record travel between routes, and Two-opt Move
with record-to-record travel. Feasibility must be maintained.
If no feasible record-to-record travel move is made, go to Step 3.
If a new record is produced, update Record and Deviation.
End I loop
**Step 3.**
For the current solution, apply One-Point Move (within and between routes),
Two-Point Move (between routes), Two-opt Move (between routes),
and Two-opt move (within and between routes).
Only downhill moves are allowed. If a new record is produced, update Record and Deviation.
**Step 4.**
Repeat Steps 2 and 3 for K consecutive iterations.
If no new record is produced, go to Step 5.
Otherwise, go to Step 2.
**Step 5.**
Perturb the solution.
Compare the solution generated after perturbation to the best solution
generated so far and keep the better solution.
Stop.

Figure 2.5: *SDVRP heuristic: Endpoint mixed integer program followed by variable-length neighbor list record-to-record travel algorithm.*

given by Dror and Trudeau (1989). The demand for customer $i$ in scenario [$\alpha$ - $\beta$] with a vehicle capacity of $k$ units is randomly selected from a uniform distribution on the interval [$\alpha k$, $\beta k$]. For each size problem, we solve 30 instances for each of the six scenarios with EMIP+VRTR. We use ILOG CPLEX 9.0 with Visual C++ (v6.0) to solve the EMIPs. Our experiments are carried out on a PC with a 1.7 GHz Pentium 4 processor and 512 MB of RAM.

The results of our experiments are given in Table 2.1. For each scenario, the EMIP+VRTR result is the median value of the solutions from 30 instances. Recall that, in the chapter by Archetti, Hertz, and Speranza (2006), SPLITABU -DT was run five times on one instance of each scenario and it clearly outperformed Dror and Trudeau's procedure (DT). Archetti (2005) provided us with the results for SPLITABU-DT given in Table 2.1(the actual problem instances were not available). These results were generated on a PC with a 2.4 GHz Pentium 4 processor and 256 MB of RAM.

It is not a straightforward task to compare the results given in Table 2.1. For each problem size and each scenario, EMIP+VRTR solves 30 instances while SPLITABU-DT solves one instance five times. In order to compare the results, we propose the following statistical test. If EMIP+VRTR and SPLITABU-DT are equally good with respect to solution quality, then SPLITABU-DT would beat the median EMIP+VRTR result about half the time. Using a binomial distribution with $n = 36$ (this corresponds to a column of SPLITABU-DT results in Table 2.1, i.e., one run over 36 cases) and $p = 1/2$, we can test the null hypothesis that the results of the two methods are equally good (H0: $p = 0.50$) against the alternative hypothesis

that SPLITABU-DT performs worse than the median value of EMIP+VRTR (Ha: $p < 0.50$).

Using a significance level of 0.01, we would reject the null hypothesis when

$$\frac{(\hat{p} - 0.5)}{\sqrt{(0.5)(0.5)/36}} \leq -2.33$$

or $\hat{p} \leq 0.3058$. In words, if SPLITABU-DT performs better than the median value of EMIP+VRTR in $(0.3058)(36) = 11$ instances or less for a single run over 36 cases, then we would reject the null hypothesis.

In Table 2.1, we point out that, for scenarios with small customer demands, most of the savings are attributable to VRTR. In the first two or three scenarios, the greater emphasis is on routing the vehicles. For scenarios with large customer demands, most of the savings are attributable to EMIP. The last three scenarios emphasize packing the vehicles. For example (using the median solutions throughout), in scenario 1 (small demand) of the 50-node problems, the EMIP solution (E2) averages a 0.61% savings over the Clarke-and-Wright solution, while the final solution after VRTR averages an 8.11% savings over E2. In scenario 6 (large demand), the EMIP solution (E2) averages a 13.89% savings over the Clarke-and-Wright solution, while the final solution after VRTR averages a 0.60% savings over E2.

For each of the five runs of SPLITABU-DT over the 36 cases, we count the number of times the SPLITABU-DT solution is better (i.e., less) than the median solution of EMIP+VRTR and provide these counts in the bottom row of Table 2.1. For each run, the count for SPLITABU-DT is much less than 11 (the counts are 4, 5, or 6) and, therefore, we reject the null hypothesis and conclude that SPLITABU-DT

performs worse than EMIP+VRTR.

In Table 2.2, we give the average running times for EMIP+VRTR and SPLITABU-DT (Archetti (2005) provided the running times for each of the five runs) on the six benchmark problems. Recall that, in EMIP+VRTR, we need to specify the values of two parameters - the size of the neighbor list (NL) and the time limit in seconds (T). We selected values for NL and T that approximately equalized the running times of EMIP+VRTR and SPLITABU-DT (these values are provided in Figure 2.5). Clearly, for a fixed problem size, the running times of both methods increase considerably as we move from the first scenario with small customer demand ([0.01 - 0.1]) and almost no splits in the final solution to the sixth scenario with very large customer demand ([0.7 - 0.9]) and several splits in the final solution. Finally, at the suggestion of a referee, we applied EMIP+VRTR to the six problems with the original demands of the customers (in other words, we solved the standard, capacitated versions of these problems). In Table 2.3, we present the solution values and running times generated by EMIP+VRTR and SPLITABU-DT (Archetti (2005) provided these results each of which is the average of five runs). EMIP+VRTR was very fast and produced solutions that were better than the average solutions produced by SPLITABU-DT.

### 2.3.2 Five Benchmark Problems With Lower Bounds

Next, we test our heuristic on problems taken from the set of 14 randomly generated problems given by Belenguer, Martinez, and Mota (2000). We use the

50 customers with vehicle capacity 160

| Scenario | EMIP+ VRTR | SPLIT-TABU | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| $[0.01-0.1]$ | 457.21 | 464.64 | 464.64 | 466.19 | 460.79 | 462.54 |
| $[0.1-0.3]$ | 723.57 | 751.60 | 767.46 | 752.84 | 760.57 | 774.56 |
| $[0.1-0.5]$ | 943.86 | 1013.00 | 1015.15 | 997.22 | 1007.13 | 1010.86 |
| $[0.1-0.9]$ | 1408.34 | 1461.01 | 1473.29 | 1470.11 | 1443.84 | 1501.39 |
| $[0.3-0.7]$ | 1408.68 | 1507.60 | 1491.92 | 1490.73 | 1487.02 | 1507.25 |
| $[0.7-0.9]$ | 2056.01 | 2166.34 | 2174.81 | 2166.11 | 2170.43 | 2148.38 |

75 customers with vehicle capacity 140

| Scenario | EMIP+ VRTR | SPLIT-TABU | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| $[0.01-0.1]$ | 598.25 | 606.52 | 601.62 | 607.07 | 608.61 | 602.39 |
| $[0.1-0.3]$ | 1081.10 | 1092.69 | 1087.94 | 1094.34 | 1104.34 | 1097.33 |
| $[0.1-0.5]$ | 1393.53 | 1449.11 | 1449.54 | 1432.17 | 1439.00 | 1448.34 |
| $[0.1-0.9]$ | 2056.54 | 2132.02 | 2136.56 | 2136.66 | 2109.70 | 2107.22 |
| $[0.3-0.7]$ | 2112.61 | 2149.74 | 2156.29 | 2167.22 | 2159.12 | 2170.18 |
| $[0.7-0.9]$ | 3067.19 | 3181.50 | 3138.18 | 3183.61 | 3216.11 | 3183.83 |

100 customers with vehicle capacity 200

| Scenario | EMIP+ VRTR | SPLIT-TABU | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| $[0.01-0.1]$ | 651.44 | 635.89 | 665.87 | 643.41 | 641.42 | 657.11 |
| $[0.1-0.3]$ | 1414.33 | 1495.76 | 1437.96 | 1438.32 | 1482.18 | 1455.84 |
| $[0.1-0.5]$ | 1973.34 | 2043.70 | 1981.55 | 2013.69 | 2077.36 | 2033.69 |
| $[0.1-0.9]$ | 3162.22 | 3172.64 | 3061.38 | 3010.51 | 3069.86 | 3193.27 |
| $[0.3-0.7]$ | 3134.56 | 2936.92 | 3090.85 | 3154.24 | 3126.00 | 2882.13 |
| $[0.7-0.9]$ | 4779.13 | 4883.37 | 4851.42 | 4909.15 | 4773.60 | 4921.42 |

Continued on next page

120 customers with vehicle capacity 200

| Scenario | EMIP+ VRTR | SPLIT-TABU | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| $[0.01-0.1]$ | 985.17 | 1084.08 | 1085.96 | 1076.09 | 1084.34 | 1093.03 |
| $[0.1-0.3]$ | 2568.90 | 2914.58 | 2916.92 | 2927.75 | 2915.13 | 2919.19 |
| $[0.1-0.5]$ | 3687.06 | 4176.24 | 4242.15 | 4131.13 | 4160.77 | 4320.33 |
| $[0.1-0.9]$ | 6079.14 | 6636.46 | 6684.42 | 6565.91 | 6259.68 | 6773.43 |
| $[0.3-0.7]$ | 6123.96 | 6433.02 | 6746.46 | 6585.96 | 6598.34 | 6834.01 |
| $[0.7-0.9]$ | 8941.79 | 10086.02 | 10494.49 | 10399.11 | 10072.61 | 10468.19 |

150 customers with vehicle capacity 200

| Scenario | EMIP+ VRTR | SPLIT-TABU | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| $[0.01-0.1]$ | 875.16 | 899.10 | 890.67 | 887.55 | 895.46 | 882.00 |
| $[0.1-0.3]$ | 1844.96 | 1922.49 | 1915.15 | 1926.86 | 1918.83 | 1907.93 |
| $[0.1-0.5]$ | 2532.93 | 2639.28 | 2644.44 | 2632.87 | 2608.92 | 2638.08 |
| $[0.1-0.9]$ | 3945.38 | 3907.38 | 3851.04 | 384,9.74 | 4056.01 | 3884.49 |
| $[0.3-0.7]$ | 4011.74 | 4001.80 | 4098.43 | 4071.44 | 4059.75 | 3967.11 |
| $[0.7-0.9]$ | 5950.35 | 6099.87 | 6239.28 | 6215.92 | 6215.49 | 6211.25 |

199 customers with vehicle capacity 200

| Scenario | EMIP+ VRTR | SPLIT-TABU | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| $[0.01-0.1]$ | 1040.20 | 1051.61 | 1058.60 | 1060.41 | 1047.88 | 1062.87 |
| $[0.1-0.3]$ | 2258.66 | 2383.90 | 2378.06 | 2386.29 | 2389.44 | 2383.11 |
| $[0.1-0.5]$ | 3202.57 | 3298.49 | 3265.60 | 3247.32 | 3333.66 | 3277.32 |
| $[0.1-0.9]$ | 5094.61 | 4737.47 | 4902.00 | 4893.66 | 4835.13 | 4900.89 |
| $[0.3-0.7]$ | 5088.08 | 5184.25 | 5103.60 | 5001.46 | 5066.96 | 5157.95 |
| $[0.7-0.9]$ | 7207.04 | 8065.69 | 7676.12 | 8007.30 | 8022.49 | 7951.60 |

Continued on next page

| | SPLIT-TABU | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| Number of times SPLITABU-DT solution is better than the median solution of EMIP+VRTR | 5 | 4 | 5 | 6 | 4 |

Table 2.1: *Computational results for six benchmark problems. For each scenario, the EMIP+VRTR result is the median value of the solutions from 30 instances. SPLITABU-DT solves one instance of each scenario five times.*

same settings of parameters given in Table 1 and do not fine-tune EMIP+VRTR. We focus on five problems with large customer demands that were generated along the lines of scenarios 4, 5, and 6 ([0.1 - 0.9], [0.3 - 0.7], [0.7 - 0.9]) from Dror and Trudeau (1989). These problems are denoted by S51D4, S51D5, S51D6, S76D4 and S101D5, where S51D4 is a problem with 51 nodes including the depot and customer demands are randomly generated according to scenario 4. Each problem has a lower bound on the optimal solution. All problems are available online at www.uv.es/belengue/sdvrp.html.

Our results are presented in Table 2.4. EMIP+VRTR generates high-quality solutions that are 5.85% above the lower bound on average. EMIP+VRTR takes 390 seconds on average to solve these five problems. Notice that, for problem S101D5, Belenguer, Martinez, and Mota (2000) report that their cutting-plane algorithm terminated early due to memory overflow. Therefore, the bound for this problem may not be a tight lower bound. In Figure 2.6 and 2.7, we show the Clarke-and-Wright and EMIP+VRTR solutions to S51D6, respectively. The EMIP+VRTR solution has 42 routes with seven customers on three routes, 12 customers on two

50 customers with vehicle capacity 160

| Scenario | EMIP+VRTR | SPLIT-TABU |
|---|---|---|
| $[0.01 - 0.1]$ | 1.9 | 4.8 |
| $[0.1 - 0.3]$ | 3.4 | 21.8 |
| $[0.1 - 0.5]$ | 14.7 | 28.2 |
| $[0.1 - 0.9]$ | 55.4 | 60.8 |
| $[0.3 - 0.7]$ | 47.9 | 48.6 |
| $[0.7 - 0.9]$ | 135.4 | 106.0 |

75 customers with vehicle capacity 140

| Scenario | EMIP+VRTR | SPLIT-TABU |
|---|---|---|
| $[0.01 - 0.1]$ | 25.8 | 13.0 |
| $[0.1 - 0.3]$ | 57.0 | 45.4 |
| $[0.1 - 0.5]$ | 214.0 | 123.0 |
| $[0.1 - 0.9]$ | 401.1 | 193.0 |
| $[0.3 - 0.7]$ | 509.6 | 129.0 |
| $[0.7 - 0.9]$ | 811.0 | 869.0 |

100 customers with vehicle capacity 200

| Scenario | EMIP+VRTR | SPLIT-TABU |
|---|---|---|
| $[0.01 - 0.1]$ | 53.9 | 57.8 |
| $[0.1 - 0.3]$ | 126.5 | 146.0 |
| $[0.1 - 0.5]$ | 287.6 | 292.8 |
| $[0.1 - 0.9]$ | 251.2 | 259.6 |
| $[0.3 - 0.7]$ | 716.5 | 777.8 |
| $[0.7 - 0.9]$ | 1024.3 | 1004.4 |

120 customers with vehicle capacity 200

| Scenario | EMIP+VRTR | SPLIT-TABU |
|---|---|---|
| $[0.01 - 0.1]$ | 36.4 | 42.4 |
| $[0.1 - 0.3]$ | 136.4 | 142.6 |
| $[0.1 - 0.5]$ | 220.7 | 268.0 |
| $[0.1 - 0.9]$ | 722.8 | 877.8 |
| $[0.3 - 0.7]$ | 605.4 | 658.6 |
| $[0.7 - 0.9]$ | 725.4 | 1825.6 |

Continued on the next page

150 customers with vehicle capacity 200

| Scenario | EMIP+VRTR | SPLIT-TABU |
|---|---|---|
| $[0.01 - 0.1]$ | 107.8 | 172.8 |
| $[0.1 - 0.3]$ | 308.0 | 393.2 |
| $[0.1 - 0.5]$ | 630.5 | 739.2 |
| $[0.1 - 0.9]$ | 2,220.0 | 2,278.0 |
| $[0.3 - 0.7]$ | 3,028.3 | 3,008.0 |
| $[0.7 - 0.9]$ | 10,038.8 | 10,223.0 |

199 customers with vehicle capacity 200

| Scenario | EMIP+VRTR | SPLIT-TABU |
|---|---|---|
| $[0.01 - 0.1]$ | 413.4 | 525.8 |
| $[0.1 - 0.3]$ | 618.5 | 754.8 |
| $[0.1 - 0.5]$ | 1,775.7 | 2,668.0 |
| $[0.1 - 0.9]$ | 3,038.1 | 3,297.2 |
| $[0.3 - 0.7]$ | 3,035.7 | 3,565.6 |
| $[0.7 - 0.9]$ | 12,542.3 | 21,849.0 |

Table 2.2: *Average running time in seconds for EMIP+VRTR and SPLITABU-DT on six benchmark problems.*

| | EMIP+VRTR | | SPLITABU-DT | |
|---|---|---|---|---|
| Customers | Solution | Time(s) | Solution | Time(s) |
| 50 | 524.61 | 1.8 | 533.55 | 13.2 |
| 75 | 840.18 | 4.0 | 849.54 | 35.8 |
| 100 | 819.56 | 3.7 | 835.62 | 57.6 |
| 120 | 1043.18 | 5.6 | 1056.01 | 38.4 |
| 150 | 1041.99 | 10.0 | 1069.84 | 389.0 |
| 199 | 1307.40 | 18.1 | 1342.85 | 386.4 |

Table 2.3: *Computational results for six capacitated VRPs.*

| Problem | Belenguer et al. Lower Bound | EMIP+VRTR | Time (s) | % Above Lower Bound |
|---|---|---|---|---|
| S51D4 | 1520.67 | 1586.5 | 201.74 | 4.33 |
| S51D5 | 1272.86 | 1355.5 | 201.62 | 6.49 |
| S51D6 | 2113.03 | 2197.8 | 301.90 | 4.01 |
| S76D4 | 2011.64 | 2136.4 | 601.92 | 6.20 |
| S101D5 | 2630.43 | 2846.2 | 645.99 | 8.20 |

Table 2.4: *Computational results for five benchmark problems with lower bounds.*

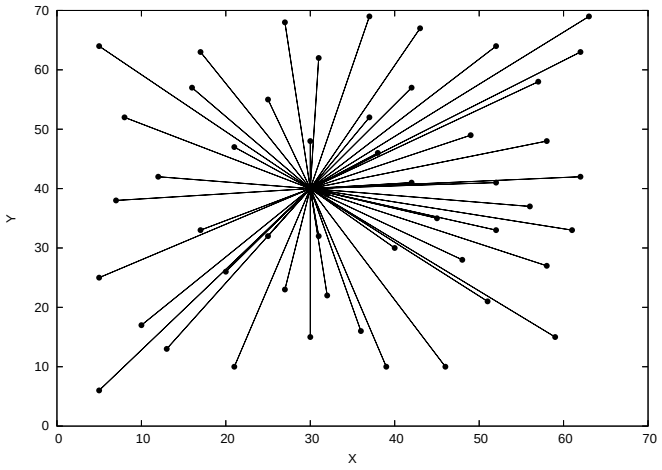



Figure 2.6: *Solutions produced by Clarke-and-Wright to problem S51D6 from Belenguer, Martinez, and Mota (2000).*

routes, and 31 customers on one route.

### 2.3.3 Twenty-one New Test Problems and Computational Results

We created 21 new test problems that range in size from 8 customers to 288 customers. Vehicle capacity is 100 units. Customer demand is either 60 units or 90 units, so our problems are generated along the lines of scenario six with very large customer demand ([0.7 - 0.9]) from Dror and Trudeau (1989). Each problem has a geometric symmetry (star shape) with customers located in concentric circles (layers) around the depot that allows us to visually estimate a near-optimal solution.

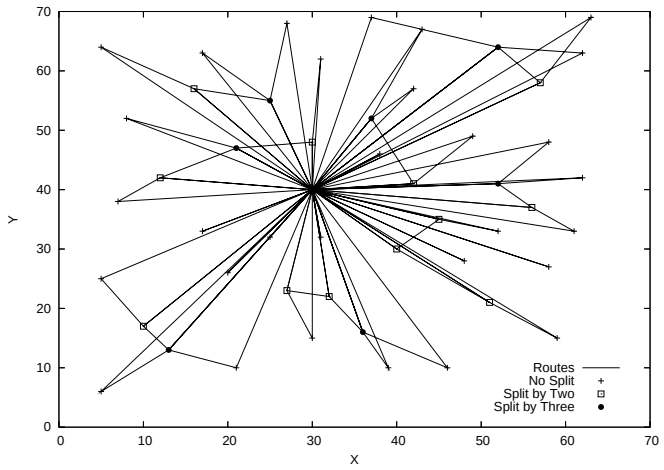



Figure 2.7: *Solutions produced by EMIP+VRTR to problem S51D6 from Belenguer, Martinez, and Mota (2000).*

We apply EMIP+VRTR to these new problems and compare the distances with the visually estimated solutions (denoted by ES). Computation times for EMIP+VRTR are also presented.

In Appendix B, we provide the problem generator and specifications for the 21 problems. We also list the data files for two problems (SD1 and SD2). (Considering the length of the dissertation we do not include all the data files. Interested readers may download them from *www.rhsmith.umd.edu/faculty/bgolden/vrp_data.htm*).

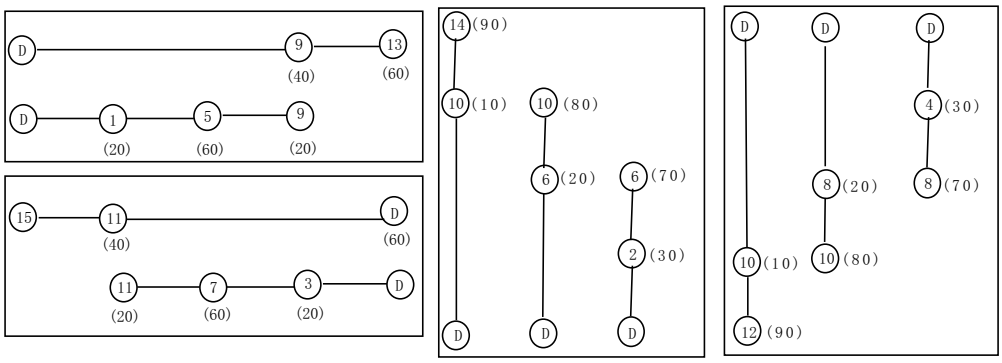
We now discuss the construction of the visually estimated solution for the new test problems. Before we do, it is useful to consider three patterns to split and consolidate the customer demands. These patterns intend to create full-truck loads (so that the vehicle capacity can be fully utilized) while keeping in mind the objective of minimizing the total traveling distance.

In the first pattern (referred to as P1), we try to use the customer demands of the same ray at first and then group the residual demands of neighboring rays.

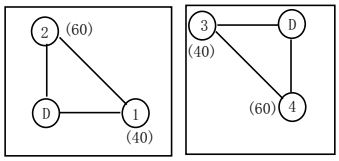
Particularly, for each ray we start with the customer farthest away from the depot and use all or partial demands of the customers along the ray to create full-truck loads. We continue this until the sum of the remaining customer demands of the same ray is less than the vehicle capacity. We then try to combine the customer demands of neighboring rays. Apply P1 to SD2, we need 12 vehicles. The solution is shown in Figure 2.8. Each square in Figure 2.8(a) contains four customers that are grouped into two/three vehicles while each square in Figure 2.8(b) contains two customers that are grouped into one vehicle. The numbers in parenthesis beside the nodes (customers) are the amount of the customer demand carried by the specific vehicle. For instance, in the top right square, four customers (1, 5, 9 and 13) are grouped into two vehicles. The first vehicle carries 60 units and 40 units from customer 13 and 9, respectively. The second one carries 20 units, 60 units and 20 units from customer 9, 5 and 1, respectively. In the right square of Figure 2.8(b), the remaining 40 units demand of customer 1 are combined with 60 units demand of customer 2 (customer 1 and customer 2 are from neighboring rays) into another vehicle.

Observe that P1 can group any five adjacent customers with demand 60 into precisely three vehicles and any ten adjacent customers with demand 90 into exactly nine vehicles.

In the second pattern (referred to as P2), two customers (with demand 60) from one particular ray are grouped with two customers (with demand 90) from its neighboring ray into three full-truck loads. Apply P2 to SD2, the solution needs 12 vehicles and is illustrated in Figure 2.9. Each square contains four customers that




(a) Routes contains customers from the same ray.




(b) Routes contains customers of neighboring rays.

Figure 2.8: *The solution to SD2 by applying P1. Each square in (a) contains four customers that are grouped into two/three vehicles. Each square in (b) contains two customers that are grouped into one vehicle.*

are grouped into three full-truck loads.

In the third pattern (referred to as P3), demands of four customers (two with demand 60 and the other two with demand 90), each from one of four neighboring layers, are grouped into three full-truck loads. Apply P3 to DS2, the solution uses 12 vehicles and is illustrated in Figure 2.10. Each square contains four customers from four neighboring rays that are grouped into three full-truck loads.

We now construct the visualized solution. Given a problem with A=4$i$ rays and B=2$j$ layers, we divide it into a set of $w = \lfloor B/10 \rfloor$ ten-layers (a ten-layer contains ten concentric circles around the depot) and the remaining $m = B \ mod \ 10$ layers. Figure 2.11 illustrates part (two rays) of the $k$th ten-layer.

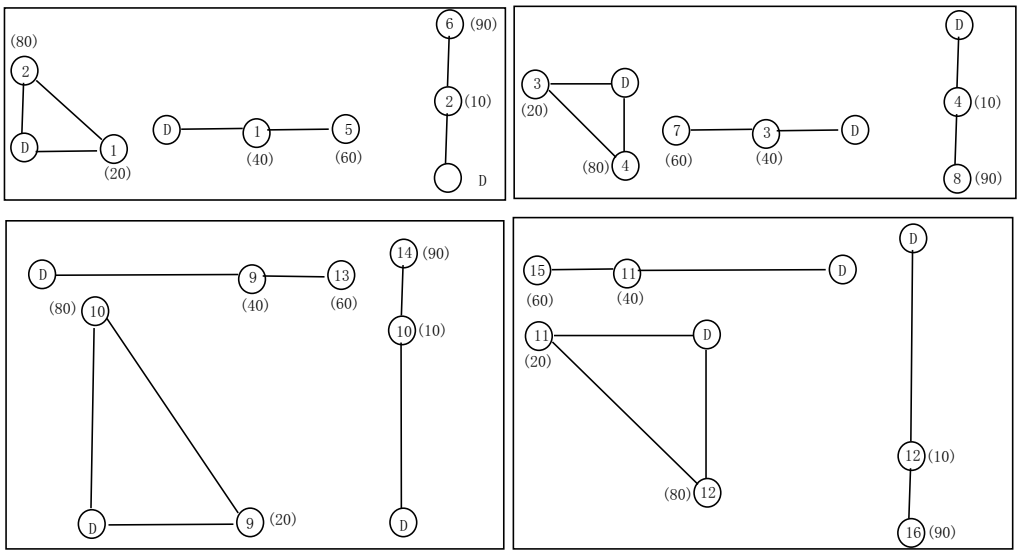



Figure 2.9: *The solution for SD2 by applying P2. Each square contains four customers (two 60-demand customers from the same ray and two 90-demand customers from two neighboring rays) that are grouped into three full-truck loads.*

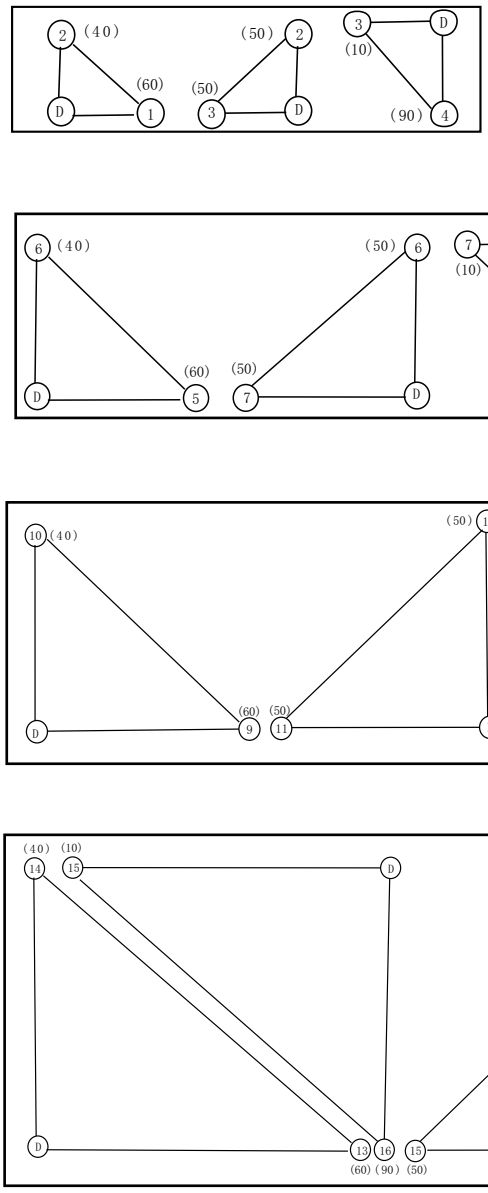



Figure 2.10: *The solution to SD2 by applying P3. Each square contains four customers from four neighboring rays that are grouped into three full-truck loads.*

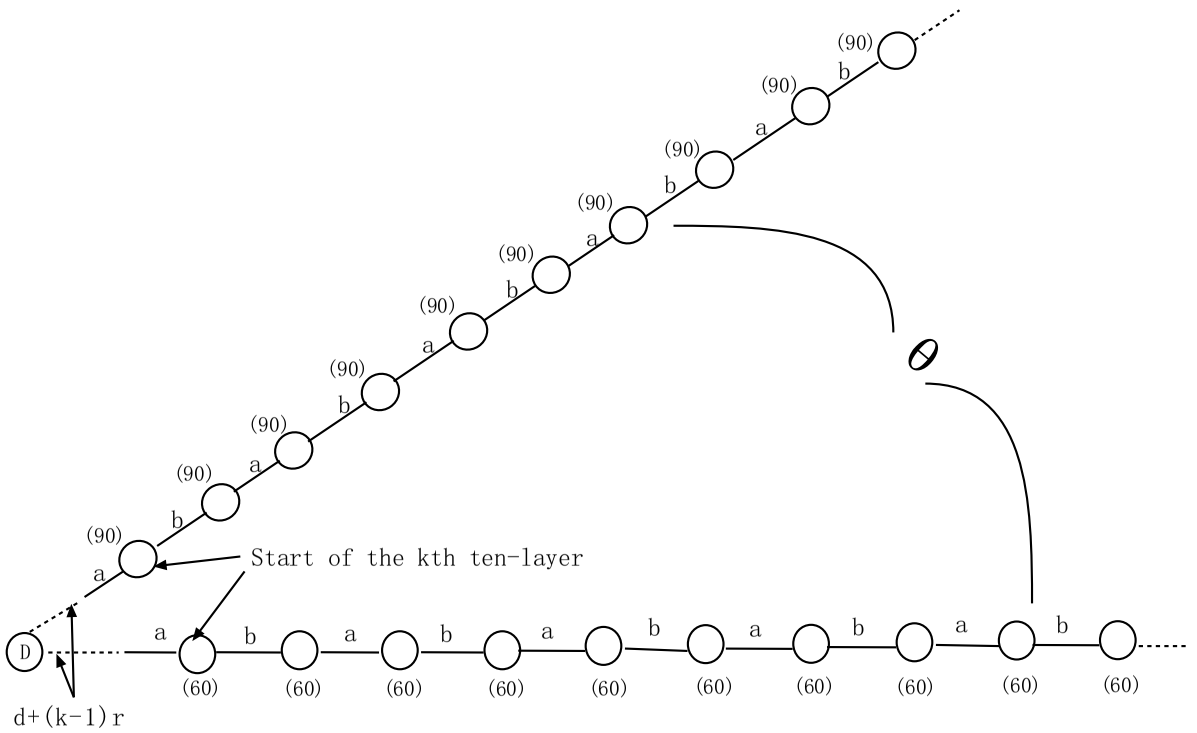



Figure 2.11: *Two rays of the kth ten-layer. The distance between two neighboring layers alternates between a and b.* $r = 5\times(a+b)$ *is the length of one ten-layer.* $d$ *is the distance from the* $m$*th layer to the depot. The distance from the depot to the start of the kth ten-layer is* $d + (k-1)r$*.* $\Theta = \frac{2\pi}{A}$ *is the angle formed by two adjacent rays.*

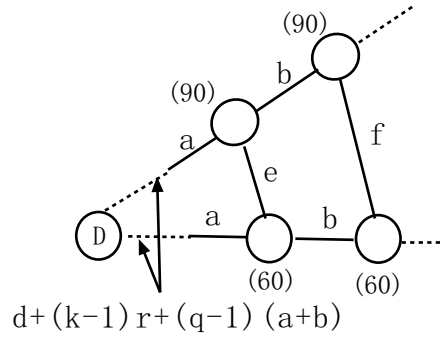



Figure 2.12: *Two rays from the qth two-layer of the kth ten-layer. The length of one two-layer is* $(a + b)$*. The distance from the depot to the start of this two-layer is* $d + (k - 1)r + (q - 1)(a + b)$*.* $e$ *and* $f$ *are the distances between two rays at the first and the second layer of the qth two-layer, respectively.*

The set of ten-layers and the remaining $m$ layers are handled differently when we construct the visualized solution. For a ten-layer, we choose from two splitting strategies described as follows.

**P1 Only:** Every ten customers in the 90-demand ray are grouped into nine full-truck loads and every ten customers in the 60-demand ray are grouped into six full-truck loads. Thus a total number of $15 \times 2i$ vehicles are needed for one ten-layer and the traveling distance is $d_{P1} = 2i\{30[(k-1)r + d] + 98a + 84b\}$.

**Mix of P2 and P3:** Further divide a ten layer into five two-layers. Figure 2.12 shows part (two rays) of the $q$th two-layer of the $k$th ten-layer. The difference in the traveling distances between P2 and P3 at the qth two-layer can be expressed as: $f_{23} = i * (2b - e - 3f) = i * \{2b - 2[(q-1)(a+b) + a]\sin(\frac{\Theta}{2}) - 6[(q-1)(a+b) + a + b]\sin(\frac{\Theta}{2})\}$.

Let $q_0 = \frac{b - 3b\sin(\frac{\Theta}{2}) - 4a\sin(\frac{\Theta}{2})}{4(a+b)\sin(\frac{\Theta}{2})} + 1$ be the break-in point at which $f_{23}$ equals zero. Observe that $f_{23}$ decreases as $q$ increases, i.e., P2 is better than P3 for $q > \lfloor q_0 \rfloor$. Specifically, if $m + 10(k-1) \leq 2(\lfloor q_0 \rfloor - 1) < 10k + m$, we choose

P3 for the layers from $m + 10(k-1)$ to $2(\lfloor q_0 \rfloor - 1)$ and choose P2 for layers from $2\lfloor q_0 \rfloor$ to $m + 10k$; If $2(\lfloor q_0 \rfloor - 1) < m + 10(k-1)$, we choose only P2; If $2(\lfloor q_0 \rfloor - 1) >= 10k + m$, we choose P3.

We compare the distances from the above two splitting strategies for each ten-layer and choose the one with a smaller traveling distance.

For the remaining $m$ layers we choose the splitting strategy according to the value of $m$. The procedure is described as follows.

- if $m = 2$, P1 and P3 are the same. Use P1(P3) if $\lfloor q_0 \rfloor \geq 1$, and P2 otherwise.
- if $m = 4$ or 6, consider the following two splitting strategies and choose the one with the shorter traveling distance.
  - P1 only.
  - If $1 \leq \lfloor q_0 \rfloor < \frac{m}{2}$ use P3 for the first $2\lfloor q_0 \rfloor$ layers and use P2 for the rest layers; If $\lfloor q_0 \rfloor < 1$, use P2 only. If $\lfloor q_0 \rfloor \geq \frac{m}{2}$, use P3 only.
- if $m = 8$, consider three splitting strategies as follows and choose the one with the shorter traveling distance.
  - Use P1 only.
  - Use P1 for the last five layers. For the remaining three layers , we compare two splitting methods as seen in Figure 2.13 and choose the better one.
  - If $1 \leq \lfloor q_0 \rfloor < 4$ use P3 for the first $2\lfloor q_0 \rfloor$ layers and use P2 for the rest layers; If $\lfloor q_0 \rfloor < 1$, use P2 only. If $\lfloor q_0 \rfloor \geq 4$, use P3 only.

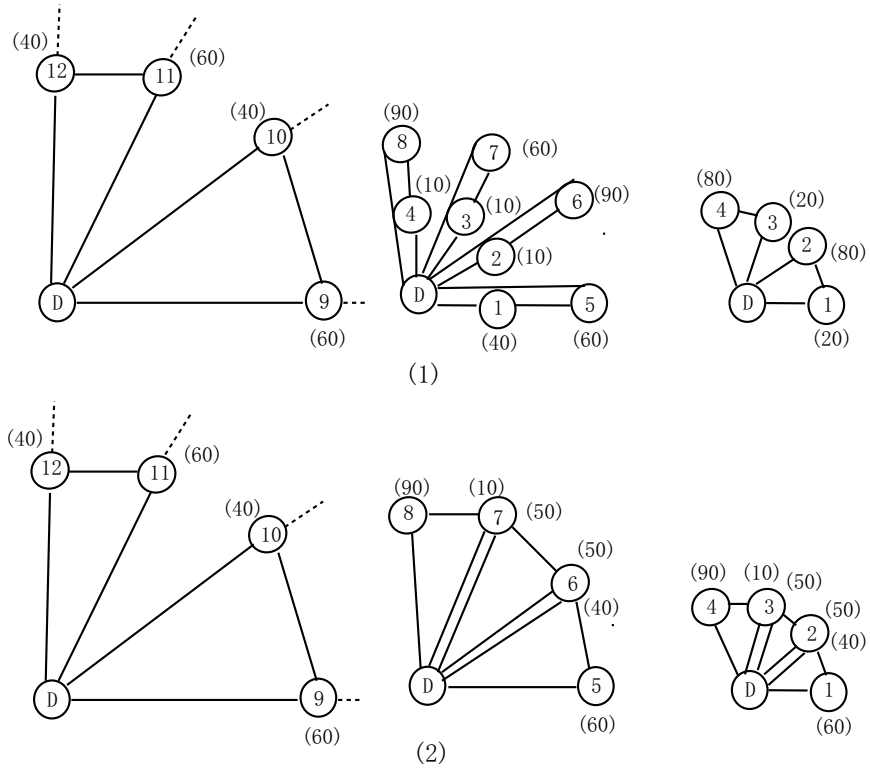



Figure 2.13: *The two splitting methods for the first three layers when m=8. Both methods use eight vehicles. The first method groups customer demands from both the same ray and different rays into full-truck loads. The second method groups customer demands of neighboring rays.*

| Problem | n | ES | EMIP+VRTR | Time (s) | % Above ES |
|---|---|---|---|---|---|
| SD1 | 8 | 228.28 | 228.28 | 0.7 | 0.00 |
| SD2 | 16 | 708.28 | 714.40 | 54.4 | 0.86 |
| SD3 | 16 | 430.61 | 430.61 | 67.3 | 0.00 |
| SD4 | 24 | 631.06 | 631.06 | 400.0 | 0.00 |
| SD5 | 32 | 1390.61 | 1408.12 | 402.7 | 1.26 |
| SD6 | 32 | 831.21 | 831.21 | 408.3 | 0.00 |
| SD7 | 40 | 3640.00 | 3714.40 | 403.2 | 2.04 |
| SD8 | 48 | 5068.28 | 5200.00 | 404.1 | 2.60 |
| SD9 | 48 | 2044.23 | 2059.84 | 404.3 | 0.76 |
| SD10 | 64 | 2684.85 | 2749.11 | 400.0 | 2.39 |
| SD11 | 80 | 13280.00 | 13612.12 | 400.1 | 2.50 |
| SD12 | 80 | 7280.00 | 7399.06 | 408.3 | 1.64 |
| SD13 | 96 | 10110.60 | 10367.06 | 404.5 | 2.54 |
| SD14 | 120 | 10920.00 | 11023.00 | 5021.7 | 0.94 |
| SD15 | 144 | 15151.10 | 15271.77 | 5042.3 | 0.80 |
| SD16 | 144 | 3381.32 | 3449.05 | 5014.7 | 2.00 |
| SD17 | 160 | 26560.00 | 26665.76 | 5023.6 | 0.40 |
| SD18 | 160 | 14380.30 | 14546.58 | 5028.6 | 1.16 |
| SD19 | 192 | 20191.20 | 20559.21 | 5034.2 | 1.82 |
| SD20 | 240 | 39840.00 | 40408.22 | 5053.0 | 1.43 |
| SD21 | 288 | 11271.10 | 11491.67 | 5051.0 | 1.96 |

Table 2.5: *Computational results for 21 new problems. ES is the total distance of the visually estimated solution. EMIP+VRTR is run once on each problem.*

In our computational experiments we set $a = b = 10$ and therefore $q_0 = \frac{1+\sin(\frac{\Theta}{2})}{8\sin(\frac{\Theta}{2})}$. We apply EMIP+VRTR to these new problems with the same settings for parameters given in Figure 2.5. In Table 2.5, we present the total distance for each problem given by the visually estimated solution (denoted by ES) and the EMIP+VRTR solution. Computation times for EMIP+VRTR are also presented.

EMIP+VRTR performs very well when compared to ES. On average, it generates solutions that are 1.29% above the near-optimal solutions generated by ES. For problems with 24 or more customers, we see that our procedure uses the maximum amount of computing time to solve the endpoint mixed integer program (400s for

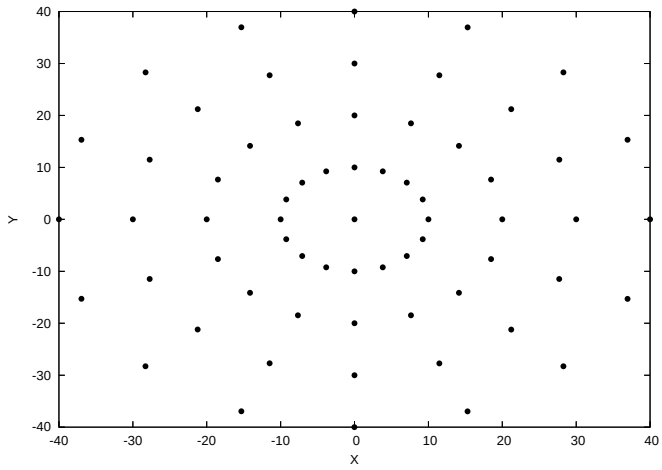



Figure 2.14: *Problem SD10 with 64 customers.*

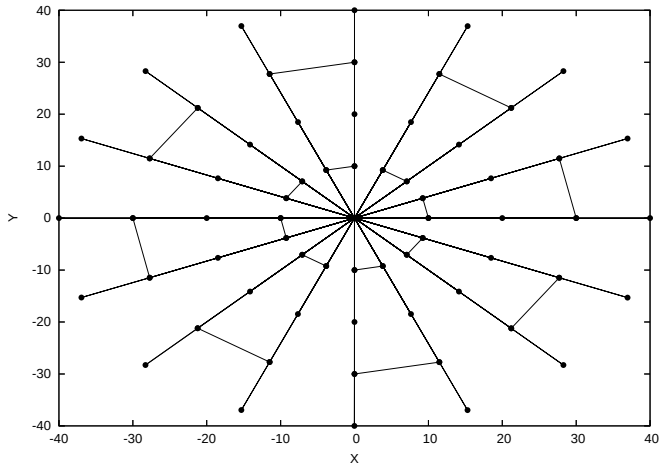



Figure 2.15: *Visually estimated solution for SD10. Total distance is 2684.85 with 48 vehicles.*

SD4 to SD13 and 5000s for SD14 to SD21). In Figure 2.14-2.16, we show the visually estimated and EMIP+VRTR solutions to SD10. The ES solution has 48 routes with 32 customers on two routes and 32 customers on one route. The EMIP+VRTR solution has 49 routes with two customers on three routes, 25 customers on two routes, and 37 customers on one route.

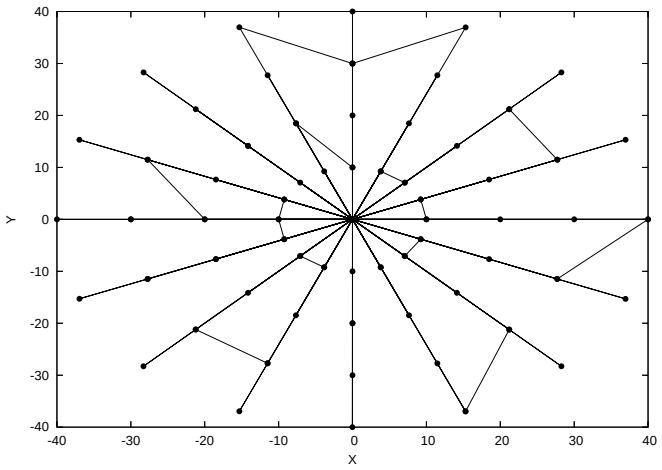



Figure 2.16: *EMIP+VRTR solution for SD10. Total distance is 2749.11 with 49 vehicles.*

## 2.4 Concluding Remarks

Over the last decade or so, practical problems that involve routing helicopters in the North Sea, distributing feed to cattle, and delivering newspapers to residential customers have been modeled using split deliveries. A variety of procedures based on mathematical programs and tabu search have been used to solve SDVRPs with some success.

We combined a mixed integer program and a record-to-record travel algorithm to form a new solution procedure (EMIP+VRTR) with only two parameters. We applied EMIP+VRTR to six benchmark problems with 50 to 199 customers and found that it consistently outperformed tabu search. On five other problems for which lower bounds exist, EMIP+VRTR again performed well. Finally, we developed 21 new problems with 8 to 288 customers that have near-optimal estimated solutions and found that our new solution procedure generated very high-quality solutions to these new problems.

In the future, we hope to refine the process of setting the values of parameters in EMIP+VRTR and to determine if some of the visually estimated solutions are optimal.

# Chapter 3

# The Regenerator Location Problem

## 3.1 Introduction

The ever-increasing usage of digital communications has inspired the development of a variety of new applications in business and consumer markets. Most recently, new services such as real-time online gaming, voice over IP (VOIP), video sharing and mobile Internet, all add a significant amount of traffic to the telecommunications networks. For instance, YouTube, the leading video-sharing site, is currently serving 100 million videos per day, with more than 65,000 videos being uploaded daily [73].

Over the years providers have built optical networks which have potentially unlimited capabilities (Mukherjee (1997)), high speed [optical carrier-192 (OC-192) has a speed of 9.952 gigbits per second (Gbps)], and low loss rate, to meet the demands of the network users. In the meantime, there have been a considerable amount of research literature on topics related to optical networking such as network architectures and infrastructure, control and management, protection and survivability. In this chapter we address a problem called the regenerator location problem (RLP), which mainly deals with one physical-layer issue in the optical networks.

As has been mentioned before, one of the advantages of optical fiber is its ability to carry signals for thousands of miles. However, the strength of the signals

deteriorate as they get farther from the source due to transmission impairments (attenuation, dispersion, cross-talk and others). In other words, the distance an optical signal may be sent without losing or falsifying the information is limited. Therefore, we need to regenerate the signals periodically using regenerators. As the cost of regenerators is very high we wish to deploy as few of them as possible, while ensuring all nodes can communicate with each other (i.e., send a signal to each other). In practice, there are three forms of signal regenerations, namely 1R (reamplicfication), 2R (reamplicfication and reshaping) and 3R (reamplicfication, reshaping and retiming) (Borella et al. (1997), Zymolka (1999) and Mukherjee (2000)). Our methodology does not distinguish between the different types of regenerators and is applicable to all of them.

Although the geographical or physical extent that a signal can travel is an important issue, it seems to have been largely ignored by the academic literature on telecommunication network design. In our literature search, we have only come across two papers Gouveia et al. (2003) and Yetginer et al. (2003) that discuss the issue of regenerator placement within the context of a large network design problem. In this chapter we consider the regenerator location problem (RLP) as a stand alone problem. We prove that it is NP-complete and discuss high-quality heuristics for it. Since network design in practice is typically done in a hierarchical fashion, we believe that the RLP problem should be addressed at the outset of the network design. Application of our procedures to solve RLP ensures placement of regenerators so that all nodes of the network may communicate without worry of physical impairments of the signal.

The rest of the chapter will be organized as follows. Sec. 3.2 proves that RLP is NP-Complete. Sec. 3.3 provides a mixed integer programming (MIP) formulation for RLP. We use this formulation to provide lower bounds on solutions to the RLP problem. Sec. 3.4 discusses heuristics for solving RLP. Sec. 3.5 presents a generalization of RLP. In RLP all nodes must be connected and all nodes are potential regenerator locations. In GRLP, a set of terminal nodes T must be connected and the potential regenerator locations are the set of nodes S. Here, $S \cup T = N$, $S \cap T \neq N$. Sec. 3.6 presents the computational results from our proposed heuristics and compares them with the solutions obtained from the MIP formulation. The last section summarizes the chapter.

## 3.2 NP-Completeness of RLP

Mathematically, the RLP problem can be described as follows. Given a network $G = \{N, E, D\}$, where $N$ is the set of nodes, $E$ is the set of edges, and $D$ is the associated distance matrix of edges, and a maximum distance of $d_{\max}$ that determines how far a signal can traverse before its quality deteriorates and needs to be regenerated. Determine a minimum cardinality subset of nodes $L$ such that for every pair of nodes in $N$ there exists a path in $G$ with the property that there is no subpath (i.e., a subsequence of edges on the path) with length $\geq d_{max}$ without internal regenerators (i.e., we do not include the end point of the subpath).

In this section we prove that RLP is NP-Complete.

**Theorem 1.** *The regenerator location problem is NP-complete.*

*Proof.* We consider a special case of Hitting Set Problem (HSP), which is stated as follows [32].

Instance: Collection $C$ of subsets of a finite set $S$, where $|c| = 2$ for all $c \in C$, positive integer $K \leq |S|$.

Question: Is there a subset $S' \subseteq S$ with $|S'| \leq K$ such that $S'$ contains at least one element from each subset in C?

We now construct the corresponding instance of the RLP. Create a node for every $s \in S$. Connect all nodes in $S$. For every $c = \{c_i^1, c_i^2\}$ in C, create a pair of nodes $v_i$ and $w_i$). Connect $v_i$ to node $c_i^1$ and $c_i^2$ (the elements of $c_i$), and $w_i$, $c_i^1$ and $c_i^2$. Set the length of all edges in the resulting graph to $d_{max}$.

The question is whether there is a feasible solution (a set of nodes $L$ where we place regenerators) to the RLP problem with cardinality less than or equal to $K$. Observe that in the RLP problem obtained from transforming a HSP, a feasible solution need not place a regenerator at the $v_i$ and $w_i$ nodes. This is due to the fact that the nodes representing the elements in $S$ are fully connected. Thus a feasible solution with a regenerator at a $v_i$ or $w_i$ node remains feasible when the regenerator is removed from that node. With this it is easy to observe that an instance of HSP has a "yes" answer if and only if the corresponding RLP has a "yes" answer. As this is a polynomial transformation, the decision version of the RLP is NP-complete.

□

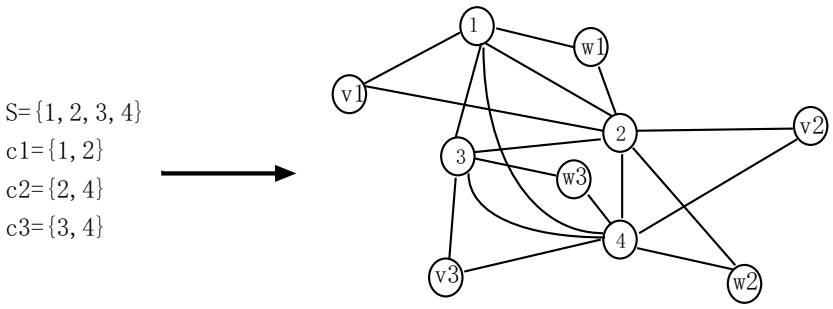



Figure 3.1: *Transform a HSP to RLP*

## 3.3 An MIP Formulation for RLP

In this section, we present a mixed integer program (MIP) formulation for RLP. Before we do, it is useful to consider the following graph transformation of RLP.

### 3.3.1 Graph Transformation

Given a graph $G = \{N, F, D\}$, and a maximum distance of $d_{max}$, apply the all pairs shortest path algorithm. Replace edge lengths by the shortest path distance. If the edge length is less than or equal to $d_{max}$ then keep the edge, and if the edge is greater than $d_{max}$ delete the edge. Denote this new graph as $M$ (with node set $N$ and edge set $E$). Observe then, if $M$ is a complete graph, no regenerators are required. On the other hand every node pair that is not connected by an edge in $M$ requires regenerators to communicate. We call such node pairs "not directly connected" or NDC node pairs. It suffices to consider the RLP problem on the transformed graph $M$ and determine the minimum cardinality subset of nodes $L$, such that for the NDC node pairs in $M$ there exists a path with regenerators at all internal nodes

on the path. Observe the following property in the graph $M$. Suppose we place a regenerator at a node $t$. Then every node pair that is not directly connected in $M$, but that is connected to $t$ can communicate. Consequently, after the placement of a regenerator at node $t$, such node pairs can be viewed as being directly connected (i.e., can communicate with each other) and the graph $M$ can be updated with edges between such nodes. In this setting, our objective then is to minimize the number of nodes where regenerators are placed so that $M$ is fully connected.

Figure 3.2-(a) illustrates a small example with four nodes. $d_{max}$ is set to 100. The numbers besides the edges are the edge distances, which are replaced by the shortest path lengths in Figure 3.2-(b). Notice that edge (a,c), (a,d) and (b,d) are longer than 100, and thereby are removed in the new graph $M$ as shown in Figure 3.2-(c). A possible regenerator deployment that can connect all the nodes is to add two regenerators, one at node b and the other at node c. Observe, with this, there exists a path between every pair of nodes where a signal does not travel more than $d_{max}$ before being regenerated.

### 3.3.2 The MIP Formulation

To formulate RLP as an MIP, we further transform $M$ to a new graph $M'$ as follows:

- For every node $i$ in $M$, create two nodes $i_1$ and $i_2$ in $M'$, and add an arc $< i_1, i_2 >$ in $M'$ with cost $c'_{i_1,i_2} = 1$.
- For every edge $(i, j)$ in $M$, add two arcs $< i_2, j_1 >$ and $< j_2, i_1 >$ in $M'$ and

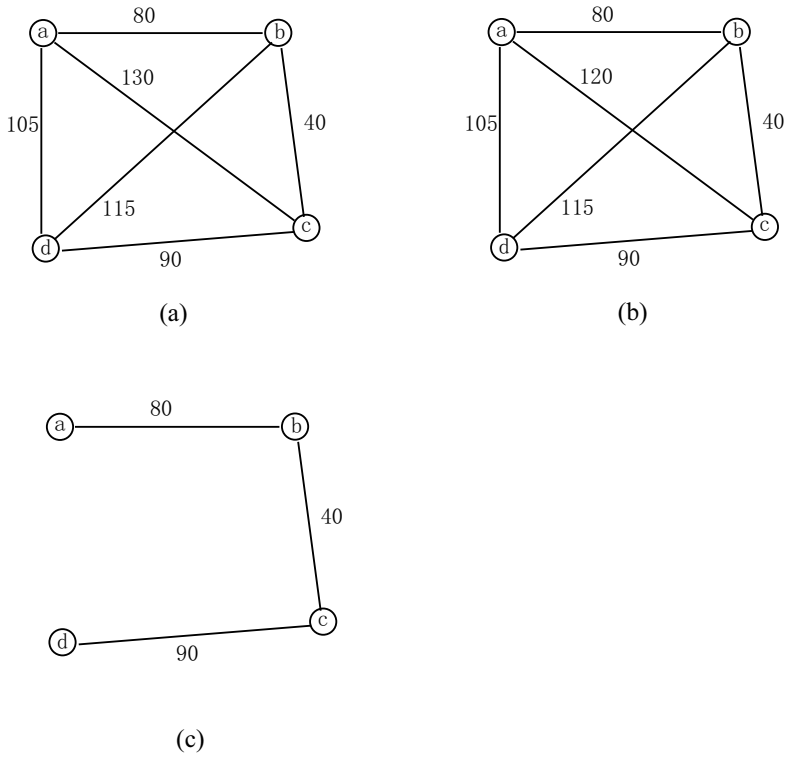



Figure 3.2: *A Small Example with four nodes and* $d_{max} = 100$

set their costs, $c'_{i_2,j_1}$ and $c'_{j_2,i_1}$ to zero;

Figure 3.3 illustrates the transformation for Figure 3.2-(c). We now model RLP as an uncapacitated fixed-charge network design problem. For every node pair $(i,j)$ in $M$ that is not directly connected we create a distinctive commodity $p$ in $M'$. We wish to send 1 unit of flow (i.e., find a path) from the origin $(o_p)$ $i_2$ to the destination $(d_p)j_1$. The fixed costs are as given in $C'$ (i.e., some arcs have zero cost, and some arcs have unit costs) and the flow costs are zero. We wish to find a minimum cost solution. To see the correspondence between RLP and the uncapacitated fixed-charge network design problem observe that, in particular, choosing an arc $< i_1, i_2 >$ in the solution to the problem on $M'$ is equivalent to installing a regenerator to node

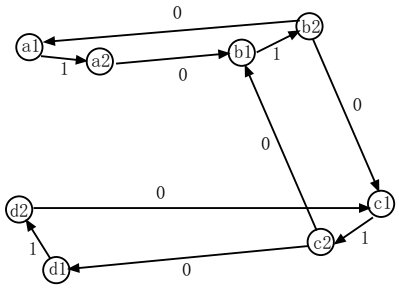



Figure 3.3: *The transformed graph $M'$ from $M$*

$i$ in $M$. Thus the problem can be viewed as a special case of the multi-commodity uncapacitated fixed charge network flow problem (MUFC) [54].

We now formulate RLP as a multi-commodity flow problem. The notations for our formulation are listed as follows:

**Notations**

$n$ total number of nodes in $M$;

$P$ the set of $(i,j)$ node pairs that are not directly connected in $M$;

$m$ the cardinality of $P$;

$N'$ the set of nodes in $M'$

$A'_1$ the set of arcs in $M'$ with cost 1.

$o_p$ the origin node of the $p$th node pair in $P$;

$d_p$ the destination node of the $p$th node pair in $P$;

**Decision Variables:**

$$x_{ij} = \begin{cases} 1, & \text{if arc} < i, j > \in A'_1 \text{ is used;} \\ 0, & \text{Otherwise.} \end{cases}$$

$f_{ij}^p$ the amount of flow sent from node $o_p$ to node $d_p$ via arc $< i, j > \in A'$

**Formulation:**

$$\min \sum_{<i,j>\in A_1'} x_{ij} \tag{3.1}$$

$$\sum_{i\in N', <i,j>\in A'} f_{ij}^p - \sum_{i\in N', <j,i>\in A'} f_{ji}^p = \begin{cases} -1, & j = o_p \\ 1, & j = d_p \\ 0, & \text{otherwise} \end{cases} \quad \forall p \in P \tag{3.2}$$

$$f_{ij}^p \leq x_{ij} \;\; \forall < i, j > \in A_1' \;\; and \;\; p \in P \tag{3.3}$$

$$f_{ij}^p \geq 0, \;\; \forall < i, j > \in A' \;\; and \;\; p \in P \tag{3.4}$$

$$x_{ij} \in B \;\; \forall < i, j > \in A_1' \tag{3.5}$$

Constraint 3.2 is the flow conservation constraint. Constraint 3.3 says if arc $< i, j >$ is used by any commodity $p$ it must be selected in the solution ($x_{ij} = 1$). Constraint 3.4 is simply the non-negativity constraint for $f_{ij}^p$ variables and constraint 3.5 limits the $x_{ij}$ variables to be binary.

## 3.4 Proposed Heuristics

In this section we discuss three heuristics, referred as Greedy, H1 and H2. Before we do, we first discuss a preprocessor (to reduce the problem size and fix regenerators in the solution ) and a post-optimizer (to improve upon the heuristic solution) that we apply to all of our heuristics.

### 3.4.1 The Preprocessor

Observe that if node $i$ in $M$ is only connected to one other node $j$, i.e., $i$ has degree one, every feasible solution must include a regenerator deployed at node $j$. Once a regenerator is deployed at node $j$, node $i$ can be eliminated from $M$. The preprocessor is described as follows.

Let $L \subseteq N$ be the set of selected regenerator locations in the solution.

1. Initialization: $L = \emptyset$, $n_1 = n_2 = 0$.

2. If there is no one-degree node, go to 4; Otherwise, let $n_1$ equal to its index and $n_2$ equal to the node connecting to it. Remove $n_1$ from $M$ ($M = M \setminus n_1$). If $n_2$ is already in $L$, go back to 2.

3. Let $D(n_2)$ be the degree of node $n_2$ in $M$.

   - If $D(n_2) = 0$ and $n_2$ is the only node left in $M$, $L$ is a feasible solution. Stop.
   - If $D(n_2) = 0$ but there are still other nodes in $M$, preprocessing is complete.
   - If $D(n_2) \geq 1$, add $n_2$ to $L$. Furthermore, if $D(n_2) = 1$ then $n_2$ is the new one-degree node. Go to 2.

4. Update the graph, i.e., remove all the NDC node pairs that can be connected after deploying the regenerators in $L$, and adding to $M$ the edges associated with such node pairs.

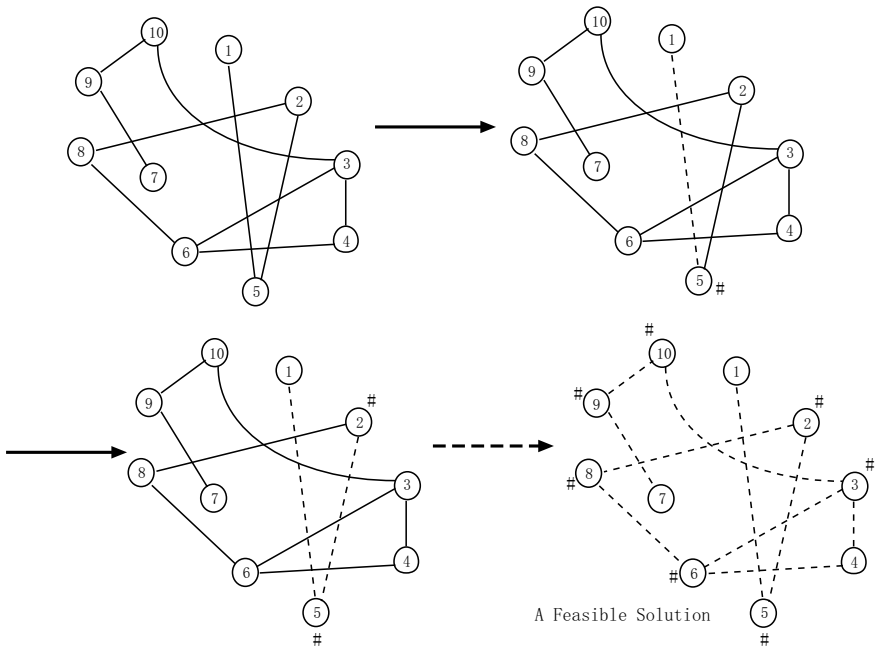



Figure 3.4: *Preprocessor Finds the Optimal Solution: Example 1*

Figure 3.4 gives an example where we obtain a feasible solution after applying the preprocessor. In the first step, the preprocessor detects that node 1 has degree one. Thus node 1 is eliminated from the graph and a regenerator is added to node 5 (the node that is connected to node 1). In the resulting graph, node 5 has degree one, so the preprocessor removes node 5 and adds a regenerator to node 2. The process continues until no such node can be found. The final regenerator deployment (nodes with # are chosen regenerator locations) in this example is feasible. Observe that the preprocessor can find a feasible solution for any graph structured as a spanning tree.

We now show the running time of the preprocessor is $O(F)+O(|N|^3)$. The data structure we adopt is the adjacency list representation. The preprocessor searches for $n_1$, eliminates it from the graph, checks the degree of $n_2$ on the reduced graph.

It repeats the above steps until (1) the degree of $n_2$ is zero which means the graph is either disconnected or is already complete, or (2) no more one-degree node can be found. Essentially, the preprocessor scans each edge in the graph which takes at most $O(F)$ time. By the end of the preprocessor, if the resulting $L$ is not yet feasible, we need to update the graph. Specifically, for every $t \in L$, all the neighbors of $t$ now communicate with each other. Accordingly, the adjacency lists of the neighbors need to be updated. The update takes $O(|N|^2)$ time to update the adjacency lists of the nodes connected to each $t \in L$ and $O(|N|^3)$ for the set $L$.

### 3.4.2 The Post-optimizer

The post optimizer consists of two subroutines: **K-swap** and **Remove**. **K-swap** tries to swap $K$ ($K$=1 or 2) regenerator locations in the current solution with locations that are not regenerator locations. If **K-swap** results in a feasible solution, we continue to apply **Remove**. **Remove** simply tries to remove a regenerator location from the current solution. Figure 3.5 illustrates an example where a combination of **1-SWAP** and **REMOVE** improves the solution. The initial solution has three regenerators deployed at nodes 5, node 8 and node 10, respectively. Observe if we remove any of the regenerator the problem is no longer feasible. However, **1-SWAP** moves one regenerator from node 5 to node 1 and then **Remove** removes a regenerator from node 10. The resulting solution has only two regenerators (at node 1 and node 8) instead of three.

Observe that **1-SWAP** takes $O(|N|^2)$, **2-SWAP** takes $O(|N|^3)$ and **Remove**

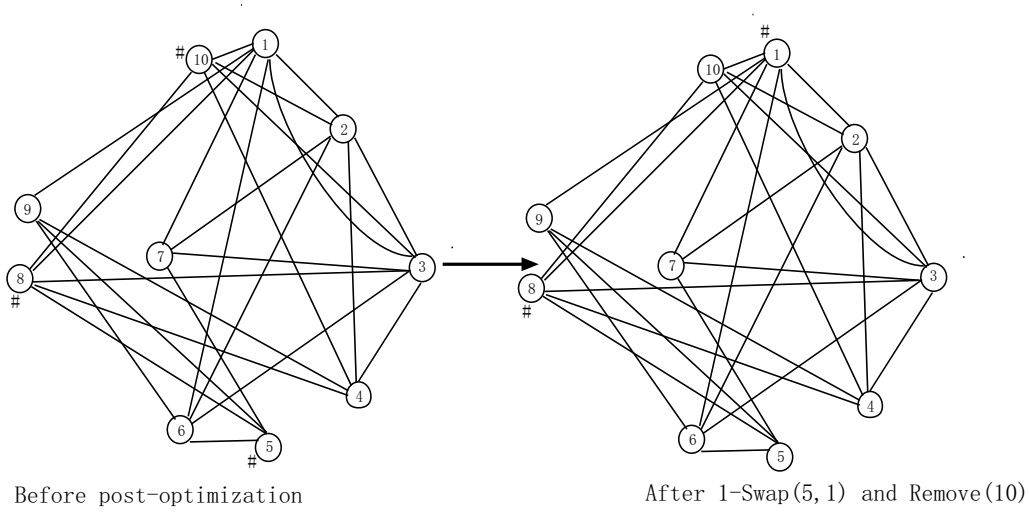



Figure 3.5: *Swap node 5 with node 1 and Remove node 10*

takes $O(|N|)$. Thus, the running time of the post-optimizer is $O(|N|^3)$.

### 3.4.3 Greedy Heuristic

The Greedy heuristic tries to find the node which can eliminate the most NDC node pairs if a regenerator is deployed at its location. It keeps looking for such nodes until a feasible solution is reached.

We discuss the complexity of the Greedy heuristic in terms of memory requirement and running time. As has been mentioned previously, we use the adjacency list representation to store the graph. This list is updated as the regenerators are added to the graph. Recall that with a feasible regenerator deployment the graph should be fully connected, which requires $O(|N|^2)$ storage space. The Greedy heuristic calculates for each node the number of NDC node pairs ($nr(i)$) it can reduce if a regenerator is added at its location. It takes $O(|N|^2)$ to find $nr(i)$ for one node and $O(|N|^3)$ for all the nodes in the graph. It also takes $O(|N|)$ comparisons to find the

one with the largest $nr(i)$. Once the node is identified, we add a regenerator to its location, update $M$, and check the feasibility (i.e., whether M is fully connected). Since we can at most add $N$ regenerators, the complexity of the Greedy heuristic is $O(|N|^4)$. Note that this is a worst-case running time bound and the actual running time is much faster.

### 3.4.4 Heuristic H1

We make three observations that will be the basis of H1 (and our second heuristic H2). First, a RLP instance is feasible only if the underlying graph is connected. Second, every connected graph has at least one spanning tree. And third, we can obtain a feasible solution from a spanning tree by deploying one regenerator to every non-leaf node. Obviously the fewer non-leaf nodes a spanning tree has the fewer regenerators are needed in the corresponding feasible solution. In H1, we use a subroutine called TREE($i$) that tries to construct a spanning tree with as few non-leaf nodes as possible.

Heuristic H1 has the following steps.

1. Initialization. $S = \emptyset$, $C_i = \emptyset$ and visit($i$)= **false** $\forall i \in N$.

2. Find the node $i$ that has the lowest degree.

3. Call $L$ = TREE($i$).

The pseudo-code for TREE($i$) is provided in Figure 3.6. Let $S \subseteq N$ be the set of the nodes that have been added to the spanning tree in construction. $F(S) \in F$

**TREE**$(i)$
{
  Visit$(i)$ = true;
  For each neighbor of $i$ that is not in $S$ or $L$, add it to the list $C_i$;
  If $S = \emptyset$, let $S = S \bigcup i$; Otherwise let $S = S \bigcup i \bigcup C_i$;
  While not all the nodes in $C_i$ are visited
  {
    Among all the unvisited nodes in $C_i$ find the node $c$ that has the largest degree, $D_c$, in the graph $\{N, F \backslash F\{S\}\}$;
    If $D_c > 0$
    {
      $L = L \bigcup c$;
      TREE$(c)$;
    }
    Else, return $L$;
  }
  Return $L$;
}

Figure 3.6: *Steps of Tree*$(i)$.

is the set of edges that have both vertices in $S$. In addition, let $L$ be the set of chosen regenerator locations and $C_i$ be node $i$'s neighboring nodes which are not in $S$ or $L$.

The intuition behind TREE$(i)$ is that if we add a regenerator to a node with a larger degree in $F \backslash F\{S\}$ (i.e., choose it as a non-leaf node), there is a better chance that we can connect more NDC node pairs.

We illustrate H1 using a small example shown in Figure 3.7. In the graph, node 1 has the lowest degree, thus H1 starts by calling TREE(1). We have $C_1 = \{2, 8\}$, $S = \{1\}$ and $F\{S\} = \emptyset$. The degrees of node 2 and node 8 in $F \backslash F\{S\}$ are both three (we call the degree of nodes in $F \backslash F\{S\}$ the revised degree). Breaking the

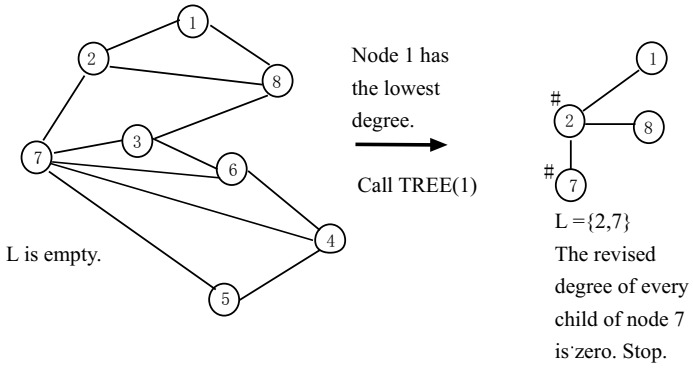



Figure 3.7: *Apply H1 to a Small Example*

tie arbitrarily, let H1 choose node 2 and call Tree(2). We have $C_2 = \{7, 8\}$ and $S = \{1, 2, 8, 7\}$. Since the revised degree of node 7 is four, which is greater than that of node 8, we call TREE(7). We have $C_7 = \{3, 6, 4, 5\}$ and $S = \{1, 2, 8, 7, 3, 6, 4, 5\}$. The revised degree of every element of $C_7$ is zero. Thus H1 stops and returns $L = \{2, 7\}$.

We now show that the running time of H1 is $O(|N| + |F|)$. It takes $O(|N|)$ comparisons to find the node with the lowest degree. H1 then calls the TREE subroutine. TREE($i$) takes $O(D_i)$ ($D_i$ is the revised degree of node $i$) time to check all the neighbors of node $i$ and adds those that are not in either $S$ or $V$ to set $C_i$. It then takes $\sum_{k=1}^{|C_i|} D_{i_k}$ ($D_{i_k}$ is the degree of the $k$th neighbor of $i$) operations to obtain the revised degree of all the nodes in $C_i$ and $O(|C_i|)$ comparisons to find the node $c$ with the largest revised degree. The subroutine continues with TREE($c$). Since there are at most $N$ nodes the running time for TREE is bounded by $O(\sum_{i=1}^{N}(\sum_{k=1}^{|C_i|} D_{i_k} + C_i)) = O(|F| + |N|)$. Thus, the running time of H1 has a complexity of $O(|N| + |F|)$.

```
Node(i)
{
   Among all the nodes adjacent to node i that are not in L
   find the node c that has the largest degree;
   Return c;
}
```

Figure 3.8: *Steps of Node*($i$).

### 3.4.5 Heuristic H2

Heuristic H2 has the following steps.

1. Find the node $i$ that has the lowest degree. If the degree equals to $|N| - 1$, the solution is feasible, go to 3.

2. Call $U = \text{Node}(i)$. $L = L \cup U$, update the graph and go to 1.

3. Return $L$.

We provide the pseudo-code for Node($i$) in Figure 3.8. The difference between H1 and H2 is that the former tries to construct the spanning tree in one shot while the latter starts all over again every time a regenerator location is selected.

We now show that the running time of H2 is $O(|N|^3)$. H2 first takes $O(|N|)$ to find the node $i$ with the lowest degree. It then calls Node($i$) which spends $O(D_i)$ time checking the neighbors of $i$ and finding the node $c$ which has the largest degree and is not yet in $L$. H2 adds a regenerator at node $c$ and spends $O(|N|^2)$ time updating the adjacency lists. Since we can at most add $N$ regenerators Node($i$) is called for at most $N$ times. Thus, the running time for H2 is bounded by $O(|N|^3)$.

## 3.5 Generalized RLP

In the section we consider the generalized regenerator location problem (GRLP) which is a generalization to RLP. In GRLP, we are given a graph $G = \{N, S, T, F, D\}$, where $N$ is the set of nodes, $S$ is the set of nodes that can host a regenerator, $T$ is the set of terminal nodes, $F$ is the set of edges and $D$ is the associated distance matrix. The relationship between $S$ and $T$ are: $S \bigcup T = N$ and $S \bigcap T \neq S \neq T$. A path needs to be established to carry traffic between each pair of nodes in $T$. The traffic can be carried for a maximum distance of $d_{max}$ before they become weak and need to be regenerated. A regenerator can only be installed at an $S$ node. The objective is to determine where to put those regenerators (a set of regenerator locations $L$) so that their total number is minimized.

Like the RLP, we obtain the graph $M$ by replacing all the edge lengths in $G$ with the shortest path distances and deleting the edges longer than $d_{max}$. We call the $T$ node pairs that are not directly connected in $M$ the NDC node pairs.

The MIP formulation for RLP can be used to solve GRLP by further transforming $M$ to a new graph $M'$ as follows:

- For every node $s$ in $S$ create two nodes $s_1$ and $s_2$ in $M'$, and add an arc $< s_1, s_2 >$ in $M'$ with cost $c'_{s_1,s_2} = 1$.

- For every node $t$ in $T$ create two nodes $t_1$ and $t_2$ in $M'$, and add an arc $< t_1, t_2 >$ in $M'$ with cost $c'_{t_1,t_2} = |N|$.

- For every edge $(i, j)$ in $M$, add two arcs $< i_2, j_1 >$ and $< j_2, i_1 >$ with costs

$$c'_{i_2,j_1} = c'_{j_2,i_1} = 0.$$

### 3.5.1 Proposed Heuristics for GRLP

In this subsection, we propose two heuristics for GRLP, referred to as GGD and GH2. The former is essentially a greedy algorithm while the latter an extension to H2. Similar to RLP, we use a preprocessor and a post-optimizer. We now describe the above procedures.

#### 3.5.1.1 The Preprocessor

The preprocessor for GRLP needs to distinguish between a pure $T$ node and an unpure $T$ node (we call a $T$ node that is also an $S$ node an unpure $T$ node), and check the $S$-degree (we call the number of $S$ nodes connected to a node its $S$-degree) of the T nodes. Specifically, if a pure $T$ node has $S$-degree one, or if an unpure $T$ node has $S$-degree one and is not fully connected to all the other $T$ nodes, we deploy a regenerator at the $S$ node $s$, i.e., add $s$ to the set of regenerator locations $L$. In addition, we eliminate the pure $T$ node from $M$. Observe that, different from the preprocessor for RLP where eliminating a node may change the degree of the remaining nodes, eliminating a pure $T$ node does not affect the $S$-degree of the remaining nodes. In other words, we only need to check all the $T$ nodes once, which takes $O(|T|)$ time. At the end of the preprocessor, if $L$ is not yet feasible, we need to update the graph. Specifically, for every $s \in L$, all the neighbors of $s$ now communicate with each other. Accordingly, the adjacency lists

of the neighbors need to be updated. The update takes $O(|N|^2)$ for each $s \in L$ and $O(|N|^2|S|)$ for the whole set $L$. Therefore, the running time for the GRLP preprocessor is $O(|T|)+O(|N|^2|S|)$.

#### 3.5.1.2 The Post-optimizer

The post-optimizer for GRLP is similar to that of RLP except that the **K-swap** and the **Remove** involve only $S$ nodes.

#### 3.5.1.3 Heuristic GGD

GGD is essentially a greedy algorithm. The basic idea is to find the $S$ node(s) that can reduce the most NDC node pairs when a regenerator is installed at it(them). Every time the algorithm identifies such a node(s), it deploys a regenerator(s) at the location(s) and updates the graph. GGD stops when all the $T$ nodes in the problem are directly connected with each other.

#### 3.5.1.4 Heuristic GH2

We describe GH2 as follows.

1. Find the node $i$ with the most NDC nodes from set $T$. Let it be the root node.

2. Perform a breadth-first search on the tree and stop at the layer where there is at least one $S$ node that connects to some $T$ node that is not directly connected to $i$ (we call it $T$-degree). Find the node $c$ that has the largest $T$-degree. Add a regenerator to $c$ and update the graph.

3. If the resulting graph is feasible, stop; Otherwise, go back to Step 1.

We demonstrate GH2 using an example with S={3,7,8} and T={1,2,4,5,6} as shown Figure 3.9(a). Among all the T nodes in the graph, node 5 and node 6 have the maximum number of NDC nodes. Arbitrarily break the tie, select node 5 as the root node. Among its adjacent $S$ nodes node 3 is connected to the most $T$ nodes that are not already connected to node 5, i.e., the largest $T$-degree. We deploy a regenerator at it. Notice that all the neighbors of node 3 can communicate with each other. We update the the graph by adding edges connecting node pairs (1, 2), (1, 6) and (2, 5)[Figure 3.9(b)]. Next, we find that node 4 has the maximum number of NDC nodes and its adjacent S nodes, node 7 and node 8, have the same $T$-degree. Arbitrarily break the tie, we deploy a regenerator at node 7. Notice that node 4 and node 5 can communicate with each other. We update the graph by adding an edge between them [Figure 3.9(c)]. In the resulting graph node 4 and node 6 have the maximum number of NDC nodes. We arbitrarily choose node 4. Among its adjacent $S$ node, node 8 has the largest $T$-degree. We add a regenerator at node 8 [Figure 3.9(d)]. Figure 3.10 illustrates the sequence of node scan.

## 3.6 Computational Results

We now discuss our computational experience with the five heuristics. We compare our heuristics with the exact solution obtained by applying CPLEX 9.0 (a commercial MIP solver) to an MIP formulation of the problem. For the small-sized instances we are able to obtain the optimal solutions. For the remaining instances,

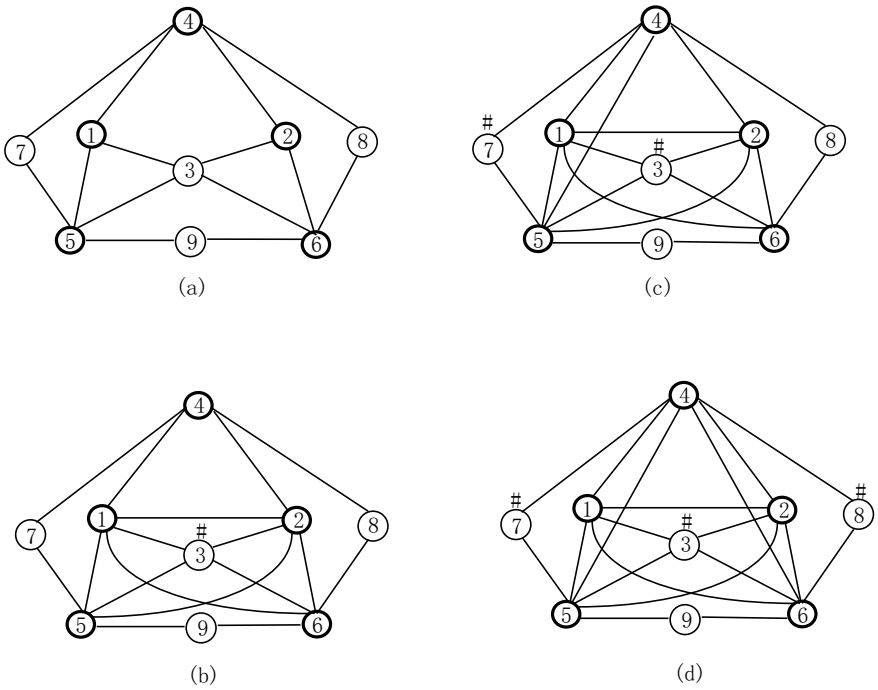



Figure 3.9: *The example with S={3,7,8} and T={1,2,4,5,6}*

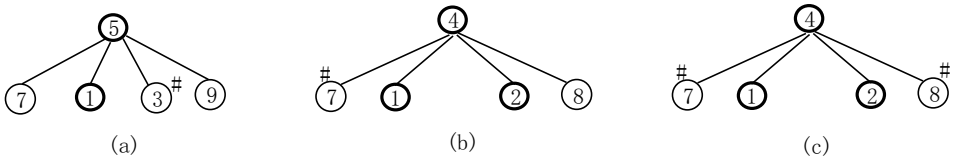



Figure 3.10: *Node Scan for the example with S={3,7,8} and T={1,2,4,5,6}*

we allow the CPLEX MIP solver to run for 7,000 seconds and record the best lower bounds. All the computational experiments are conducted on a PC with Pentium IV 3.4GHz and 1G of RAM.

In an attempt to capture the attributes of various networks, we consider three types of networks, viz.: the randomly generated networks (i.e., we generate the graph $M$ directly), networks with random distances and Euclidean networks. We describe their characteristics as follows.

### 3.6.1 Randomly Generated Networks

The randomly generated networks directly generate the transformed graph $M$ (i.e., we do not generate any distances but instead generat the graph $M$). A node pair is considered NDC if there is no edge connecting them. We now describe the generation procedure for an instance with $n$ nodes and $m$ NDC node pairs. We first construct an arbitrary spanning tree over all the nodes. The tree is then expanded with $n(n-1)/2-(n-1)-m$ randomly generated edges. Observe that since a complete network over $n$ nodes have $n(n-1)/2$ edges and a spanning tree has $(n-1)$ edges, there are $m$ NDC node pairs in the resulting network. For our computational experiments we generate seven problem sets with sizes range from 40 to 100 nodes. Each set has four instances with different $m$ (randomly generated).

### 3.6.2 Networks with Random Distances

The networks with random distances (and the networks with Euclidean distances) generate networks and edge lengths. The graph $M$ is then computed from these randomly generated networks. We start from a complete graph and assign length to the edges based on parameters p, a and b. Parameter p controls the percentage of edges with distances greater than $d_{max}$. Parameter a and b define the uniform distribution $[a\% \times d_{max}, b\% \times d_{max}]$ from which we randomly choose the edge length for the remaining $(1-p\%)$ percent edges. We use Floyd-Warshall algorithm [5] to calculate the all-pair shortest path for the graph. Edge lengths are then replaced with the shortest path distances. Edges longer than $d_{max}$ are eliminated. To

ensure feasibility of the instances, we randomly generate a spanning tree over all the nodes. We generate 21 sets of such networks with sizes range from 40 to 100 nodes. For each size of networks, we use three sets of parameters: $(p = 80, a = 25, b = 75)$, $(p = 90, a = 1, b = 100)$ and $(p = 90, a = 25, b = 75)$. Four instances are generated for each set of parameters. $d_{max}$ is 100 for all the instances.

### 3.6.3 Euclidean Networks

In the Euclidean networks, nodes are randomly located on a $100 \times 100$ square grid. Euclidean distances are used as edge lengths. Similar to the networks with random distances, the graphs are transformed into $M$. We generate 21 problem sets with sizes from 40 nodes to 100 nodes. For each problem size we generate four instances for each $d_{max}$ =30, 40 or 50).

### 3.6.4 Computational Analysis

We now discuss the performances of the heuristics. First, we explain the notations in the Tables. Column "$n$" is the number of the nodes. The outputs from CPLEX are recorded in columns "MIP". "LB" is the lower bound on the number of regenerators [1]. An asterisk beside the number indicates an optimal solution. "RT" records the running time in second. If the solver terminates due to the 7,000 seconds time limit we leave"RT" empty. "BP" is the number of regenerators provided by the heuristics before the post-optimization procedure. "NF" is the number of regenerators after the post-optimization procedure. The last column "Diff" is calculated

[1]We actually round up the bound produced by CPLEX

by comparing the best solution from H1, H2 and GD with the lower bound.

Table 3.1-3.3 summarize the computational results for the three types of networks, respectively. There are totally 49 problem sets in the tables. Each row in the table corresponds to one problem set containing four instances. For each instance we take the best solution from the three heuristics and compare it with the lower bound provided by the MIP model. Observe that CPLEX succeeds in finding the optimal solutions for eight problem sets, for which our heuristics also find the optimal solutions but at a much faster pace. For the remaining problem sets, we compare the heuristic solutions against the lower bounds. In Table 3.1 and 3.2, Diff ranges from 0 to 1.75. In Table 3.3, the Diff ranges from 0 to 6.75. The big difference occurs when CPLEX fails to even solve the LP relaxation of the problem within the time limit and therefore a value of one is used as the lower bound.

In general, instances with large $m$ values require longer running time. We think the reason is that more NDC node pairs need to be connected. For the randomly generated networks $m$ is explicit. In the networks with randomly generated distances, $m$ is decided by the parameter set $(p, a, b)$. Observe that if we keep $d_{max}$ unchanged, the larger $p$, $a$ and $b$ are the larger $m$ is. For the Euclidean networks, a smaller $d_{max}$ leads to a larger $m$.

| n | MIP | | H2 | | | GD | | | H1 | | | Diff |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LB | RT | BF | NF | RT | BF | NF | RT | BF | NF | RT | |
| 40 | 4.25 | – | 5.00 | 4.50 | 0.50 | 4.75 | 4.50 | 1.00 | 5.00 | 4.75 | 0.50 | 0.00 |
| 50 | 6.25 | – | 8.25 | 8.00 | 3.25 | 7.75 | 7.75 | 5.50 | 9.00 | 7.50 | 2.00 | 1.00 |
| 60 | 6.50 | – | 9.25 | 8.25 | 11.00 | 9.25 | 8.25 | 12.25 | 11.00 | 8.75 | 7.25 | 1.50 |
| 70 | 11.00 | – | 13.75 | 13.00 | 46.50 | 13.00 | 12.75 | 48.25 | 18.00 | 13.50 | 34.00 | 1.50 |
| 80 | 10.75 | – | 14.25 | 13.25 | 121.50 | 13.00 | 12.75 | 111.25 | 16.25 | 13.25 | 63.00 | 1.75 |
| 90 | 7.25 | – | 9.75 | 9.25 | 70.75 | 9.25 | 8.50 | 56.00 | 12.00 | 8.75 | 98.25 | 1.25 |
| 100 | 8.50 | – | 10.75 | 10.50 | 175.00 | 10.25 | 10.00 | 189.00 | 14.00 | 10.50 | 151.50 | 1.50 |

Table 3.1: *Computational Results for Randomly Generated Networks*

| | | | | MIP | | H2 | | | GD | | | H1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n | p | a | b | LB | RT | BP | NF | RT | BP | NF | RT | BP | NF | RT | Diff |
| 40 | 80 | 25 | 75 | 2.00* | 973.45 | 2.00 | 2.00 | 0.00 | 2.00 | 2.00 | 0.00 | 2.00 | 2.00 | 0.00 | 0.00 |
| | 90 | 1 | 100 | 5.50* | 732.25 | 5.50 | 5.50 | 1.00 | 5.50 | 5.50 | 0.00 | 6.50 | 5.50 | 1.00 | 0.00 |
| | | 25 | 75 | 4.25* | 133.33 | 4.50 | 4.25 | 0.50 | 4.50 | 4.25 | 0.00 | 5.25 | 4.25 | 0.50 | 0.00 |
| 50 | 80 | 25 | 75 | 2.00* | 998.55 | 2.00 | 2.00 | 0.00 | 2.00 | 2.00 | 0.25 | 2.50 | 2.00 | 0.00 | 0.00 |
| | 90 | 1 | 100 | 3.50 | – | 3.75 | 3.50 | 0.50 | 4.00 | 3.50 | 1.00 | 5.00 | 3.50 | 0.75 | 0.00 |
| | | 25 | 75 | 5.00 | – | 5.50 | 5.50 | 0.75 | 5.50 | 5.25 | 2.25 | 7.00 | 5.50 | 1.75 | 0.25 |
| 60 | 80 | 25 | 75 | 1.50 | – | 1.50 | 1.50 | 0.00 | 1.50 | 1.50 | 0.00 | 1.50 | 1.50 | 0.25 | 0.00 |
| | 90 | 1 | 100 | 3.00* | 1385.70 | 3.33 | 3.00 | 0.67 | 3.33 | 3.00 | 1.33 | 3.00 | 3.00 | 0.33 | 0.00 |
| | | 25 | 75 | 4.00 | – | 4.00 | 4.00 | 1.33 | 4.33 | 4.00 | 2.67 | 5.00 | 4.00 | 1.67 | 0.00 |
| 70 | 80 | 25 | 75 | 1.50 | – | 1.50 | 1.50 | 0.25 | 2.00 | 1.50 | 0.25 | 1.50 | 1.50 | 0.50 | 0.00 |
| | 90 | 1 | 100 | 2.00 | – | 2.00 | 2.00 | 0.67 | 2.33 | 2.00 | 0.67 | 2.00 | 2.00 | 0.33 | 0.00 |
| | | 25 | 75 | 3.50 | – | 4.00 | 4.00 | 1.50 | 4.50 | 4.00 | 7.25 | 4.75 | 4.25 | 4.00 | 0.50 |
| 80 | 80 | 25 | 75 | 1.00 | – | 1.00 | 1.00 | 0.00 | 2.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| | 90 | 1 | 100 | 1.00 | – | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.33 | 1.00 | 1.00 | 0.00 | 0.00 |
| | | 25 | 75 | 3.00 | – | 4.67 | 4.33 | 6.00 | 4.33 | 4.00 | 6.67 | 5.00 | 4.33 | 4.67 | 1.00 |
| 90 | 80 | 25 | 75 | 1.00 | – | 1.00 | 1.00 | 0.00 | 2.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| | 90 | 1 | 100 | 1.00* | 32.03 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.33 | 1.00 | 1.00 | 0.00 | 0.00 |
| | | 25 | 75 | 3.25 | – | 3.75 | 3.75 | 7.00 | 4.00 | 3.50 | 13.00 | 4.75 | 4.00 | 11.25 | 0.25 |
| 100 | 80 | 25 | 75 | 1.25 | – | 1.25 | 1.25 | 0.25 | 1.50 | 1.25 | 1.25 | 1.25 | 1.25 | 0.25 | 0.00 |
| | 90 | 1 | 100 | 1.00* | 38.80 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.33 | 1.00 | 1.00 | 0.00 | 0.00 |
| | | 25 | 75 | 2.25 | – | 3.75 | 3.50 | 6.75 | 4.25 | 3.00 | 14.00 | 4.00 | 3.50 | 7.00 | 0.75 |

Table 3.2: *Computational Results for Networks with Random Distances*

| n | $d_{max}$ | MIP | | H2 | | | GD | | | H1 | | | Diff |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | LB | RT | BP | NF | RT | BP | NF | RT | BP | NF | RT | |
| 40 | 30 | 7.75* | 4048.08 | 7.75 | 7.75 | 0.50 | 8.00 | 8.00 | 1.00 | 8.00 | 8.00 | 0.75 | 0.00 |
| | 40 | 4.50 | – | 4.50 | 4.50 | 0.50 | 5.25 | 4.50 | 0.50 | 5.25 | 4.50 | 0.00 | 0.00 |
| | 50 | 3.00 | – | 3.00 | 3.00 | 0.00 | 3.25 | 3.00 | 0.00 | 3.25 | 3.00 | 0.50 | 0.00 |
| 50 | 30 | 6.25 | – | 7.50 | 7.00 | 0.75 | 8.00 | 6.75 | 2.75 | 7.75 | 7.25 | 1.50 | 0.50 |
| | 40 | 4.00 | – | 4.50 | 4.50 | 0.25 | 4.75 | 4.50 | 1.00 | 5.25 | 4.75 | 0.50 | 0.25 |
| | 50 | 0.75 | – | 2.75 | 2.75 | 0.00 | 3.25 | 2.75 | 0.50 | 2.75 | 2.75 | 0.25 | 1.25 |
| 60 | 30 | 3.25 | – | 8.75 | 8.50 | 6.25 | 8.75 | 8.50 | 8.75 | 9.75 | 8.25 | 7.25 | 4.25 |
| | 40 | 2.00 | – | 4.75 | 4.50 | 1.25 | 5.25 | 4.50 | 3.50 | 5.75 | 4.75 | 1.75 | 1.75 |
| | 50 | 3.00 | – | 3.00 | 3.00 | 1.00 | 3.50 | 3.00 | 0.75 | 3.50 | 3.00 | 1.00 | 0.00 |
| 70 | 30 | 1.00 | – | 7.75 | 7.75 | 4.75 | 7.75 | 7.75 | 11.00 | 8.25 | 8.00 | 7.75 | 6.75 |
| | 40 | 1.00 | – | 5.00 | 5.00 | 2.25 | 5.25 | 5.00 | 4.25 | 6.25 | 5.00 | 1.75 | 4.00 |
| | 50 | 1.00 | – | 2.75 | 2.75 | 1.25 | 4.00 | 2.75 | 2.50 | 3.25 | 2.75 | 1.25 | 1.75 |
| 80 | 30 | 1.00 | – | 8.00 | 7.50 | 12.75 | 7.50 | 7.25 | 21.50 | 8.50 | 7.25 | 10.00 | 6.25 |
| | 40 | 1.00 | – | 4.50 | 5.50 | 3.00 | 5.50 | 4.25 | 7.25 | 5.50 | 4.25 | 4.25 | 3.25 |
| | 50 | 1.00 | – | 3.25 | 3.25 | 2.00 | 3.25 | 3.00 | 3.50 | 3.50 | 3.00 | 1.00 | 2.00 |
| 90 | 30 | 1.00 | – | 7.50 | 7.00 | 30.25 | 8.00 | 7.50 | 52.75 | 9.00 | 7.25 | 22.75 | 6.00 |
| | 40 | 1.00 | – | 4.00 | 4.00 | 3.75 | 5.50 | 4.75 | 13.50 | 5.50 | 4.50 | 9.00 | 3.00 |
| | 50 | 1.00 | – | 3.50 | 3.25 | 2.75 | 3.75 | 2.75 | 5.75 | 3.50 | 2.75 | 1.75 | 1.75 |
| 100 | 30 | 1.00 | – | 7.50 | 7.25 | 43.50 | 8.50 | 8.50 | 71.00 | 9.25 | 9.00 | 43.50 | 6.25 |
| | 40 | 1.00 | – | 4.50 | 4.50 | 7.75 | 5.25 | 4.75 | 17.00 | 5.25 | 4.25 | 11.25 | 3.25 |
| | 50 | 1.00 | – | 3.25 | 3.00 | 6.75 | 3.75 | 3.25 | 11.00 | 3.75 | 3.00 | 4.00 | 2.00 |

Table 3.3: *Computational Results for Euclidean Networks*

We also generate various problem sets for GRLP, where two more parameters $p_s$ and $p_t$ are needed. $p_s(p_t)$ specifies the percentage of the nodes that are $S(T)$ nodes. The number of S nodes $nS = p_s \times n$ and that of T nodes $nT = p_t \times n$. We mark the first $nS$ nodes in N as $S$ nodes, and the last $nT$ as $T$ nodes. We generate data for the three types of networks as mentioned previously. The problem sizes range from 40 to 100 nodes, and for each network size we use two parameter sets ($p_s = 30$, $p_t = 80$) and ($p_s = 40$, $p_t = 70$). Additionally, we use $d_{max} = 100$ and $(p = 80, a = 25, b = 75)$ for the networks with randomly generated distances, and $d_{max} = 30$ for the Euclidean networks. There are totally 42 problem sets and each has four instances. Again, to ensure feasibility, we randomly generate a spanning tree over the $S$ nodes for all the instances. Alternatively, we can ensure the feasibility by connecting the two nodes of each NDC node pair to at least one connected $S$ component (we call a connected subgraph of $M$ which only consists of $S$ nodes an $S$ component). The computational results are provided in Tables 3.4 to 3.6. We compare the better of the two heuristic solutions against the CPLEX outputs. Observe that CPLEX is able to find the optimal solutions for 29 problem sets, out of which the best of our heuristic solutions matches 23. For the remaining 4 problem sets Diff range from 0.5 to 0.75. Over the entire experiment, Diff range from 0 to 6. Again, the large gap occur where one is used as the lower bound.

Overall, the computational results demonstrate that our heuristics can find the optimal solutions for small-sized problems much faster than the exact algorithm. In addition, it can rapidly find solutions to large-sized problems where CPLEX fails to find the optimal solutions.

| n | nS | nT | MIP | | GH2 | | | GGD | | | Diff |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | NF | RT | BP | NF | RT | BP | NF | RT | |
| 40 | 12 | 32 | 5.25* | 4.48 | 5.25 | 5.25 | 0.00 | 5.75 | 5.25 | 0.50 | 0.00 |
| | 16 | 28 | 4.06* | 12.47 | 4.81 | 4.31 | 0.00 | 5.19 | 4.06 | 0.38 | 0.00 |
| 50 | 15 | 40 | 5.00* | 26.40 | 7.00 | 6.00 | 0.25 | 7.00 | 5.75 | 1.00 | 0.75 |
| | 20 | 35 | 4.50* | 663.23 | 5.00 | 4.75 | 0.75 | 5.25 | 4.50 | 0.50 | 0.00 |
| 60 | 18 | 48 | 5.00* | 1042.45 | 6.25 | 5.50 | 0.75 | 6.00 | 5.50 | 2.50 | 0.50 |
| | 24 | 42 | 4.75* | 1705.18 | 6.00 | 5.25 | 1.25 | 6.25 | 5.25 | 2.50 | 0.50 |
| 70 | 21 | 56 | 5.50* | 4872.05 | 6.50 | 6.50 | 1.25 | 6.75 | 6.00 | 5.25 | 0.50 |
| | 28 | 49 | 4.75 | – | 5.50 | 5.25 | 1.75 | 5.25 | 5.00 | 4.25 | 0.25 |
| 80 | 24 | 64 | 2.50 | – | 6.25 | 6.00 | 4.50 | 6.25 | 6.00 | 9.25 | 2.75 |
| | 32 | 56 | 1.00 | – | 5.50 | 5.50 | 2.50 | 6.25 | 5.50 | 11.50 | 3.50 |
| 90 | 27 | 72 | 1.00 | – | 6.00 | 5.75 | 5.75 | 7.00 | 5.75 | 18.25 | 4.75 |
| | 36 | 63 | 1.00 | – | 6.75 | 5.50 | 17.00 | 6.00 | 6.00 | 18.25 | 4.50 |
| 100 | 30 | 80 | 1.00 | – | 6.25 | 6.00 | 13.75 | 7.50 | 6.00 | 28.75 | 5.00 |
| | 40 | 70 | 1.00 | – | 7.25 | 5.75 | 19.50 | 6.25 | 5.75 | 34.50 | 4.75 |

Table 3.4: *Computational Results for GRLP on Randomly Generated Networks*

## 3.7 Conclusions

In this chapter, we address a regenerator location problem in the optical networks. In an optical network, the distance a signal can travel is limited due to various transmission impairments. Therefore regenerators need to be placed at selected nodes to improve the quality of the transmission. Considering the high cost of the regenerators we wish to deploy as few of them as possible. We introduce a mixed integer program that produces optimal solutions for small-sized networks. We propose three heuristics that can solve large-sized RLP problems. In the generalization of RLP, we describe GRLP where the set of the candidate regenerator locations does not necessarily coincide with the set of the terminal nodes. We develop two heuristics for solving GRLP.

We conduct our computational experiments on three different types of networks including the randomly generated networks, networks with random distances

| | | | MIP | | GH2 | | | GGD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| n | nS | nT | NF | RT | BP | NF | RT | BP | NF | RT | Diff |
| 40 | 12 | 32 | 1.00* | 0.93 | 1.25 | 1.00 | 0.00 | 1.25 | 1.00 | 0.00 | 0.00 |
| | 16 | 28 | 1.00* | 0.45 | 1.00 | 1.00 | 0.00 | 1.50 | 1.00 | 0.00 | 0.00 |
| 50 | 15 | 40 | 2.00* | 1.78 | 3.00 | 2.00 | 0.00 | 2.50 | 2.00 | 0.25 | 0.00 |
| | 20 | 35 | 1.00* | 0.55 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| 60 | 18 | 48 | 1.75* | 7.95 | 1.75 | 1.75 | 0.00 | 2.25 | 1.75 | 0.50 | 0.00 |
| | 24 | 42 | 1.00* | 0.47 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| 70 | 21 | 56 | 3.50* | 361.68 | 4.00 | 3.50 | 1.00 | 4.00 | 3.75 | 2.75 | 0.00 |
| | 28 | 49 | 1.00* | 0.38 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| 80 | 24 | 64 | 3.25* | 3335.60 | 3.50 | 3.25 | 1.50 | 4.00 | 3.25 | 4.50 | 0.00 |
| | 32 | 56 | 1.00* | 0.45 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| 90 | 27 | 72 | 0.25 | – | 3.75 | 3.25 | 6.75 | 3.25 | 3.00 | 5.00 | 2.00 |
| | 36 | 63 | 1.00* | 0.40 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 |
| 100 | 30 | 80 | 1.00 | – | 4.25 | 3.75 | 4.25 | 4.25 | 3.50 | 12.50 | 2.50 |
| | 40 | 70 | 1.00* | 2.00 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 |

Table 3.5: *Computational Results for GRLP on Networks with Random Distances*

and Euclidean networks. The sizes of the networks range from 40 nodes to 100 nodes. For the small-sized networks we are able to obtain the optimal solutions using CPLEX MIP slover. This enables us to compare our heuristic solutions with the optimal solutions. For large problems, we compare the heuristic solutions with the lower bounds provided by CPLEX. Overall, the comparison shows that our proposed heuristics can rapidly find high quality solutions for the RLP and GRLP problem. As part of the future work we wish to find tighter lower bounds that can better evaluate the performances of our heuristics.

| | | | MIP | | GH2 | | | GGD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| n | nS | nT | NF | RT | BP | NF | RT | BP | NF | RT | Diff |
| 40 | 12 | 32 | 6.00* | 4.05 | 7.00 | 6.50 | 0.50 | 6.00 | 6.00 | 0.00 | 0.00 |
| | 16 | 28 | 6.00* | 9.50 | 7.00 | 6.00 | 0.00 | 6.00 | 6.00 | 1.00 | 0.00 |
| 50 | 15 | 40 | 8.00* | 8.10 | 8.00 | 8.00 | 0.00 | 9.00 | 8.00 | 0.00 | 0.00 |
| | 20 | 35 | 5.50* | 75.10 | 6.50 | 6.00 | 0.50 | 7.00 | 7.00 | 1.00 | 0.50 |
| 60 | 18 | 48 | 6.00* | 188.70 | 8.00 | 7.00 | 2.00 | 6.00 | 6.00 | 2.00 | 0.00 |
| | 24 | 42 | 6.00* | 873.10 | 7.00 | 7.00 | 2.00 | 9.00 | 6.00 | 6.00 | 0.00 |
| 70 | 21 | 56 | 7.00* | 1599.95 | 9.50 | 7.00 | 5.00 | 8.00 | 7.50 | 7.50 | 0.00 |
| | 28 | 49 | 7.00* | 2602.40 | 8.00 | 8.00 | 3.00 | 8.00 | 7.00 | 9.00 | 0.00 |
| 80 | 24 | 64 | 7.00* | – | 9.00 | 7.00 | 10.00 | 9.00 | 7.00 | 15.00 | 0.00 |
| | 32 | 56 | 6.00* | – | 7.00 | 6.50 | 6.50 | 7.50 | 6.50 | 18.00 | 0.50 |
| 90 | 27 | 72 | 1.00 | – | 8.00 | 6.00 | 7.00 | 8.00 | 7.00 | 21.00 | 4.00 |
| | 36 | 63 | 5.50 | – | 6.00 | 6.00 | 5.50 | 6.50 | 6.00 | 16.00 | 0.50 |
| 100 | 30 | 80 | 1.00 | – | 10.00 | 8.00 | 19.00 | 8.00 | 7.00 | 36.00 | 4.00 |
| | 40 | 70 | 1.00 | – | 8.00 | 8.00 | 15.00 | 7.00 | 6.00 | 32.00 | 4.00 |

Table 3.6: *Computational Results for GRLP on Euclidean Networks*

# Chapter 4

# Parametric Uncapacitated Network Design on Directed Series-Parallel Graphs

## 4.1 Introduction

In this chapter, we study the parametric uncapacitated network design problem on directed-series-parallel graphs, which has potential application in supply chain management. A directed series-parallel graph is a directed acyclic graph (a graph that contains no directed cycles) that can be constructed solely by series and parallel operations starting from a single arc. Ward (1999) describes a polynomial-time dynamic-programming algorithm for solving the minimum-aggregate-concave-cost multi-commodity flow problem on directed series-parallel graphs. Her algorithm uses the decomposition tree (AND/OR tree) data structure developed by Valdes et al. (1982) to represent the directed series-parallel graph. Wald considers problems with a single source and multiple sinks, and concave single-criterion cost functions. However, in reality firms nowadays are often faced with several conflicting criteria or objectives when making a decision. Under such circumstances, the goal becomes finding the best possible solution which still satisfies the conflicting objectives. An optimization problem must then be solved, with multiple objectives and constraints taken into consideration. This type of problem is known as either a multi-objective,

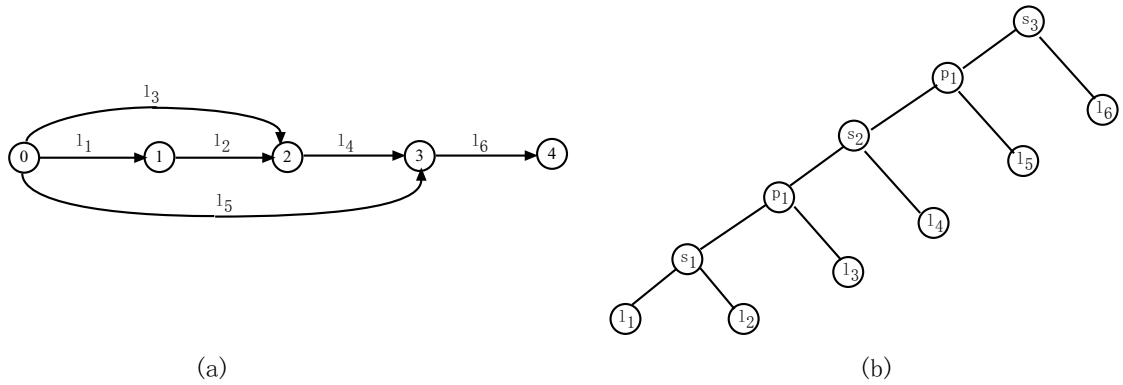
multi-criteria, or a vector optimization problem. The efficient, or Pareto optimal, solutions to such a problem is a set of solutions with the following property: in moving from one solution to the other, any improvement in one of the objective functions from its current value would cause at least one of the other objective functions to deteriorate from its current value (Frisch (1966) and Intriligator (1971)). Raghavan et al. (2002) discusses a bi-criteria product design optimization problem on AND/OR trees which can be framed as a single-source-and-single-sink network design problem on a directed series-parallel graph. The authors model the problem as the parametric (objective) optimization problem. This is achieved by transforming the bi-objective into a single objective function using a parameter $\lambda$ ranging from 0 to 1. When $\lambda = 0$, the objective function represents one objective, and when $\lambda = 1$ it represents the other objective. They develop a solution algorithm that requires as a subroutine the solution of the parametric shortest path problem. Later, Gen and Lin (2004) propose a new multi-objective hybrid genetic algorithm (moGA) approach for the bi-criteria network design problem on general graphs.

This chapter extends upon the existing literature in the sense that we consider the bi-criteria network design problem on directed series-parallel graphs with single source and multiple sinks. The rest of our chapter is organized as follows. In Sec. 4.2, we give an application of the parametric uncapacitated network design problem on directed series-parallel graphs. We then describe a polynomial-time algorithm for the problem in Sec. 4.3. In the next two sections, we discuss two special cases where the objective functions consist of (i) only linear components, or (ii) only fixed-charge components. We analyze the complexity of our algorithm in Sec. 4.6. Finally, we

summarize the contributions of this chapter in Sec. 4.7.

## 4.2 An Application of the Parametric Uncapacitated Network Design Problem on Directed Series-Parallel Graphs

Economic lot-sizing problem is one of the most fundamental problems in supply chain management. In the single-commodity uncapacitated economic lot-sizing problem there is a demand for a single commodity in each of $n$ time periods. The demand in a certain period can be satisfied by production in that period or inventory left from previous periods. Suppose there is no inventory at the beginning of the first period and no inventory is left at the end of the last period, the firm must determine a production plan so as to meet the need of each period while satisfying two opposing objectives: (i) low operational cost, and (ii) high product quality. The two objectives can be expressed as functions of the production level and inventory level. The problem can be formulated as a bi-objective single-commodity uncapacitated flow problem on a directed series-parallel graph. Figure 4.1(a) illustrates a small example with only three periods. The flow from period 0 to period $i$ represents the production level for that period. Flows between periods other than period 0 are the inventory carried on from the previous periods. The objective is to find a production plan (paths in the graph) satisfying the needs of each period (node in the graph) while meeting the two opposing objectives. Let $f$ be the production/inventory level, $C_a(f)$ be the operational cost, and $D_a(f)$ be the loss from producing inferior-quality products. We can model this problem as a parametric uncapacitated network design




(a) (b)

Figure 4.1: *Directed series-parallel graph representation of a Three-Period economic lot-sizing problem*

problem on a directed series-parallel graph by combining the two objectives into a single objective: $\lambda C_a(f) + (1-\lambda)D_a(f)$. Figure 4.1(b) is the corresponding decomposition tree of the problem. Here, the node $s_3$ contains two children $p_1$ and $l_6$. $p_1$ represents the subgraph containing nodes 0 to 3 and the arcs between them in the original series-parallel graph while $l_6$ represents the subgraph containing the arc $l_6$. Notice that, a path from node 0 to node 4 must transverse $p_1$ **and** $l_2$. Thus, we refer to $s_3$ as a series/and node in the decomposition tree. The subgraph represented by $p_1$ can be decomposed further into two subgraphs: one contains nodes 0 to 3 and the arcs between them except for $l_5$, while the other one contains the arc $l_5$. Each subgraph contains at least one alternative path from node 0 to node 3. Thus, $p_1$ is called a parallel/or node. It has two children where node $s_2$ represents the former subgraph while node $l_5$ represents the latter subgraph.

## 4.3 Preliminaries

Before we present the algorithm, it is useful to describe some notations.

**Parameters**

$\varrho$ the directed series/parallel graph

$\Im$ the series/parallel tree (decomposition tree)

$G$ a node in $\Im$, which represents a directed series-parallel subgraph

$L_G$ lower bound of G

$\sigma_G$ source node of G

$\tau_G$ tail node of G

$l_G$ left children node of G in $\Im$

$r_G$ right children node of G in $\Im$

$D_G$ the sum of demands at the internal nodes of G

$D_{\tau_G}$ demand of the tail node of G

$f_G$ a feasible flow for G

**Sets**

$A$ set of arcs

$N$ set of nodes

$V(\Im)$ the set of all the nodes in $\Im$

$\ell_{\Im}$ the set of leaf nodes of $\Im$

$\alpha_G$ the set of extreme flows of G

**Variables**

$f_G$ the flow into G

$f_G^*$ the optimal flow into G

The following concepts and lemmas are the building blocks of for our proposed algorithm. They are from Wald (1999) and Raghavan et al. (2002). Readers interested in proofs of these results should refer to these two papers.

**Concepts**

- Regular Representation. Call a network-flow problem representation regular if for every $G$ in its decomposition tree, the lower bound $L_G$ satisfy: $L_G = L_{r_G} + L_{l_G}$ if G is a p-node or $L_G = \max(L_{l_G}, D_{l_G} + D_{\tau_{l_G}} + L_{r_G})$ if G is a s-node (Wald (1999)). In Figure 4.2(a), the subgraph G represents an s-node and $L_S = \max(L_1, D_1 + D_k + L_2) = 5$. In Figure 4.2(b), the subgraph G represents a p-node and $L_G = L_1 + L_2 = 3$. If the lower bounds for all the leaf nodes are zero (notice that an arc with a positive lower bound $l_{\ell_i}$ can be replaced by a dummy node with demand $l_{\ell_i}$), the regular lower bound for a subgraph G is simply the sum of its internal demands (excluding the demand of the tail node of G).

- Extreme Flows. A feasible solution $f = (f_G)$ of a regular subgraph-flow problem is extreme if and only if for each parallel node G in the decomposition tree $\Im$, $f_g > L_g$ for at most one child g of G. Figure 4.3(a) shows a parallel subgraph G with three children, each of which has a nonnegative lower bound. The tail node of G has a demand of 3 units. A feasible solution for G has $f_G = 3$. One feasible extreme solution, as shown in Figure 4.3(b), has $f_1 = 3 > L_1$, $f_2 = L_2 = 0$ and $f_3 = L_3 = 0$. The other two feasible extreme solutions are shown in Figure 4.3(c) and Figure 4.3(d).

**Lemmas**

**Lemma 1.** *Every subnetwork-flow problem has an equivalent regular representation and the internal demands and regular sub-network lower bounds can be computed in polynomial time (Wald (1999)).*

**Lemma 2.** *If a minimum-concave-cost subgraph-flow problem is feasible, then it has an optimal solution that is extreme (Wald (1999)).*

**Lemma 3.** *The sum of two piecewise linear and concave functions, say $r_i(\cdot)$ with breakpoints $B_i$, and $r_j(\cdot)$ with breakpoints $B_j$, is also a piecewise linear and concave function. Furthermore, the number of breakpoints in the resulting function is at most $|(B_i)| + |(B_j)| - 2$, and this summation can be carried out in $O(|(B_i)| + |(B_j)|)$ time (Raghavan et al. (2002)).*

**Lemma 4.** *The minimum (lower envelope) of two piecewise linear and concave functions, say $r_i(\cdot)$ with breakpoints $B_i$, and $r_j(\cdot)$ with breakpoints $B_j$, is also a*

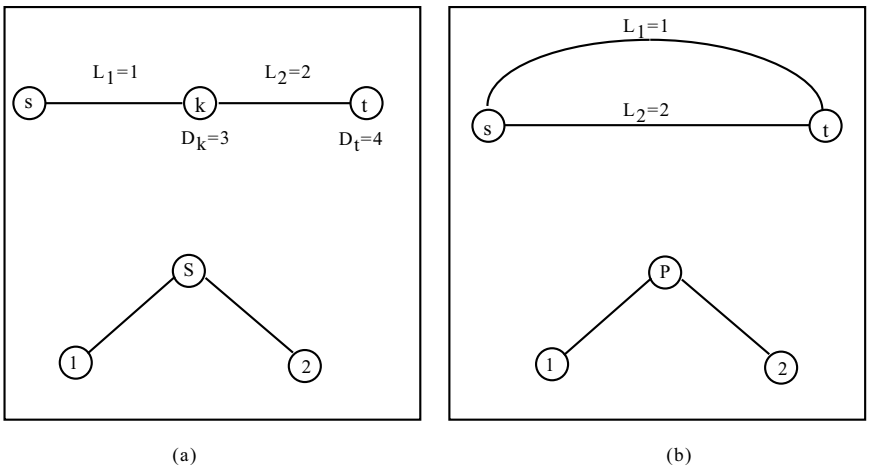



Figure 4.2: *Regular Representation*

*piecewise linear and concave function. Furthermore, the number of breakpoints in the resulting function is at most* $|(B_i)|+|(B_j)|-1$*, and this summation can be carried out in* $O(|(B_i)|+|(B_j)|)$ *time (Raghavan et al. (2002)).*

In this chapter, we assume that there are no capacity constraints on the arcs flows.

Without loss of generality, we consider problems that have regular subgraph-flow representation (Ward (1999) provides a polynomial-time transformation to regular subgraph-flow representation), and lower bound zero on the arc flows. Note that in case some arc $< i, j >$ has a positive lower bound on its flow, we can create a dummy node $k$ with demand equal to the lower bound and replace $< i, j >$ with $< i, k >$ and $< k, j >$. Furthermore, let the cost of sending flow through $< i, k >$ equal to $C_{<i,j>}$ and that of $< k, j >$ be zero. Observe that this operation retains the feature of the series and parallel graph.

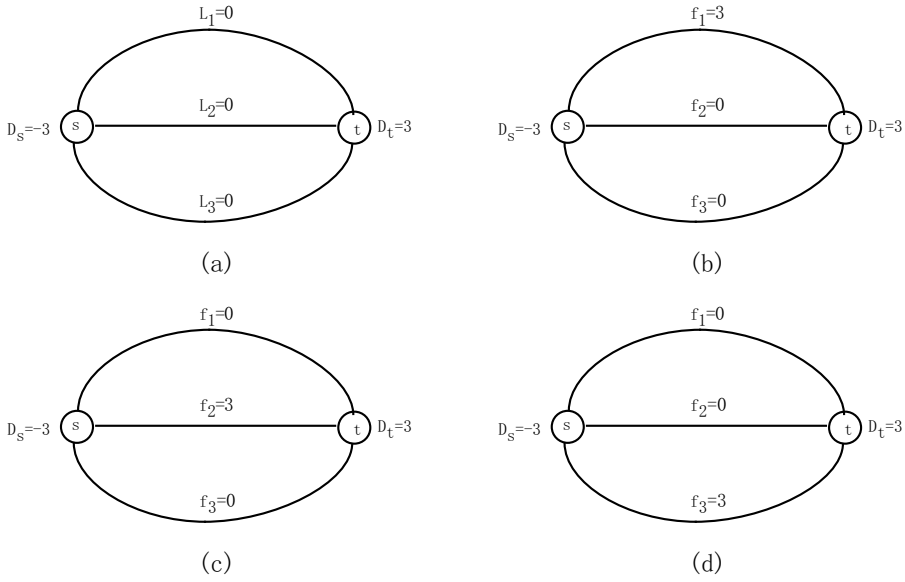



Figure 4.3: *Illustration of Extreme Flow*

## 4.4 Algorithm for The General Case (ALF)

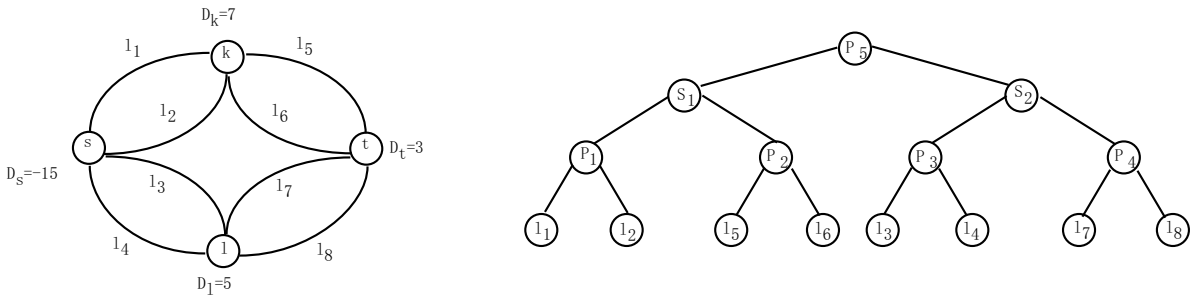
In this section we address the general case where the cost functions have both linear and fixed-cost components. We can express the transformed single-objective function in the following form:

$$C_{\ell_i} = \begin{cases} 0 & \text{if } f = 0 \\ \lambda(a_i + b_i f) + (1-\lambda)(c_i + d_i f) & \text{if } f > 0 \end{cases}$$

Figure 4.4 describes our algorithm. We use an example in Figure 4.5 to illustrate the algorithm. In Figure 4.5(a), the origin $s$ has a supply of 15 units. Node $k$ and node $l$ have a demand of 7 units and 5 units, respectively. The sink $t$ has a demand

1. algorithm ALF: **begin**
2. Construct the binary decomposition tree for the directed series-parallel graph;
3. From the bottom to the top, compute the lower bounds for each node in the tree and the demand at the top $D_{top}$;
4. From the top to the bottom, identify the extreme flow set of each node in the tree;
5. From the bottom of the tree to the top,
   for each leaf node, calculate $C_{\ell_i} = C(F)$ , $\forall F \in \alpha(\ell_i)$;
   for each s-node, calculate $C_{s_i} = C_{l_{s_i}}(F) + C_{r_{s_i}}(F - D_{l_{s_i}} - D_{\tau_{l_{s_i}}})$
   $\forall F \in \alpha(s_i)$
   for each p-node, let $C_{p_i}$ equal to the minimum of
   $C_{l_{p_i}}(F - L_{r_{p_i}}) + C_{r_{p_i}}(L_{r_{p_i}})$ and $C_{l_{p_i}}(L_{l_{p_i}}) + C_{r_{p_i}}(F - L_{l_{p_i}})$
   $\forall F \in \alpha(p_i)$;
   Record the the child node that gives the minimum cost and the corresponding breakpoint interval;
6. Construct the optimal solution for each $\lambda \in [0,1]$.
   From the top of the tree,
   at the root node, $f_r^* = L_r^* + D_{\tau_r}$;
   for each s-node, $f_{l_{s_i}}^* = f_{s_i}^*$ and $f_{r_{s_i}}^* = f_{s_i}^* - D_{l_{s_i}} - D_{\tau_{l_{s_i}}}$;
   for each p-node, find the cheaper child node corresponding to $\lambda$ and set its flow equal to $f_{p_i}^*$ minus the lower bound of the other child node, which itself carries the its lower bound;
7. **end** ALF;

Figure 4.4: *ALF: algorithm for problems with both fixed and linear objective functions*




(a) (b)

Figure 4.5: *A Simple Example*

of 3 units. The cost functions for the arcs are listed as follows.

$$C_{\ell_1}(f) = \begin{cases} 0 & \text{if } f = 0 \\ \lambda(4+3f) + (1-\lambda)(5+2f) & \text{if } f > 0 \end{cases}$$

$$C_{\ell_2}(f) = \begin{cases} 0 & \text{if } f = 0 \\ \lambda(6+2f) + (1-\lambda)(9+2f) & \text{if } f > 0 \end{cases}$$

$$C_{\ell_3}(f) = \begin{cases} 0 & \text{if } f = 0 \\ \lambda(9+4f) + (1-\lambda)(2+f) & \text{if } f > 0 \end{cases}$$

$$C_{\ell_4}(f) = \begin{cases} 0 & \text{if } f = 0 \\ \lambda(1+4f) + (1-\lambda)(7+3f) & \text{if } f > 0 \end{cases}$$

$$C_{\ell_5}(f) = \begin{cases} 0 & \text{if } f = 0 \\ \lambda(2+3f) + (1-\lambda)(8+2f) & \text{if } f > 0 \end{cases}$$

$$C_{\ell_6}(f) = \begin{cases} 0 & \text{if } f = 0 \\ \lambda(4+f) + (1-\lambda)(3+6f) & \text{if } f > 0 \end{cases}$$

$$C_{\ell_7}(f) = \begin{cases} 0 & \text{if } f = 0 \\ \lambda(7+2f) + (1-\lambda)(2+5f) & \text{if } f > 0 \end{cases}$$

$$C_{\ell_8}(f) = \begin{cases} 0 & \text{if } f = 0 \\ \lambda(1+7f) + (1-\lambda)(3+f) & \text{if } f > 0 \end{cases}$$

We construct the decomposition tree of the graph in Figure 4.5(b). Leaf nodes of the tree represent arcs on the series-parallel graph. Each p node of the tree denotes a parallel operation while each s node a series operation.

#### 4.4.0.1 Calculating Lower Bounds for Nodes in the Decomposition Tree

From the bottom of the tree, we compute the lower bound $L_G$ for each node. For all the leaf nodes, $L_G = 0$. The recursive process for p-nodes and s-nodes is as follows:

- If G is a p-node, $L_G = L_{l_G} + L_{r_G}$.
- If G is an s-node, $L_G = \max(L_{l_G}, D_{l_G} + D_{\tau_{l_G}} + L_{r_G})$, where $\tau_{l_G}$ is the tail node of the subgraph $l_G$.

The process stops at the top of the decomposition tree. The lower bounds for our example are:

$L_{\ell_1} = L_{\ell_2} = ... = L_{\ell_7} = L_{\ell_8} = 0$

$L_{p_1} = L_{\ell_1} + L_{\ell_2} = 0$; $L_{p_2} = L_{p_3} = L_{p_4} = 0$

$L_{s_1} = \max(0, 0 + 7 + 0) = 7$; $L_{s_2} = \max(0, 0 + 5 + 0) = 5$

$L_{p_5} = L_{s_1} + L_{s_2} = 12$.

The flow into the graph needs to be at least $f_{p_5} = L_{p_5} + D_{\tau_{p_5}} = 12 + 3 = 15$, which can be satisfied by the supply of $s$. Therefore, we know that this problem is feasible. The next step is to calculate the costs for each node in the tree.

#### 4.4.0.2 Computing extreme flow sets and costs

Let $f_i^k$ be the $k$th extreme flow of node $i$. From the top of the decomposition tree (Figure 4.5(b)) to the bottom, we identify the extreme flows of each node. At

the root node, the extreme flow set contains one element, which is the flow needed to satisfy the demand of the tree. In this example, $\alpha(p_5) = \{f_{p_5}^1\} = \{15\}$.

For each extreme flow in the extreme flow set of a p-node, we need to compare the cost of either (i) sending $f_p^k - L_{r_p}$ through its left child and $L_{r_p}$ through its right child or (ii) $f_p^k - L_{l_p}$ through its right child and $L_{l_p}$ through its left child. In other words, for each $f_p^k$ two extreme flows are derived for each of its two children, in particular, $f_p^k - L_{r_p}$ and $L_{l_p}$ for its left child, and $f_p^k - L_{l_p}$ and $L_{r_p}$ for its right. We now can write the extreme flow sets for its two child nodes as $\alpha_{l_p} = (f_p^1 - L_{r_p}, ..., f_p^k - L_{r_p}, L_{l_p})$ and $\alpha_{r_p} = (f_p^1 - L_{l_p}, ..., f_p^k - L_{l_p}, L_{r_p})$, respectively.

At an s-node, the extreme flow sets for its two child nodes are obvious: $\alpha_{l_s} = (f_s^1, ..., f_s^k)$ and $\alpha_{r_s} = (f_s^1 - D_s, ..., f_s^k - D_s)$, respectively.

The extreme flow sets for the rest of the nodes in our example are listed as follows:

$\alpha(s_1) = \{f_{s_1}^1, f_{s_1}^2\} = \{L_{s_1}, f_{p_5} - L_{s_2}\} = \{7, 10\}$

$\alpha(s_2) = \{f_{s_2}^1, f_{s_2}^2\} = \{L_{s_2}, f_{p_5} - L_{s_1}\} = \{5, 8\}$

$\alpha(p_1) = \{f_{p_1}^1, f_{p_1}^2\} = \{f_{s_1}^1, f_{s_1}^2\} = \{7, 10\}$

$\alpha(p_2) = \{f_{p_2}^1, f_{p_2}^2\} = \{f_{s_1}^1 - D_{p_1} - D_{\tau_{p_1}}, f_{s_1}^2 - D_{p_1} - D_{\tau_{p_1}}\} = \{0, 3\}$

$\alpha(p_3) = \{f_{p_3}^1, f_{p_3}^2\} = \{f_{s_2}^1, f_{s_2}^2\} = \{5, 8\}$

$\alpha(p_4) = \{f_{p_4}^1, f_{p_4}^2\} = \{f_{s_1}^1 - D_{p_3} - D_{\tau_{p_3}}, f_{s_1}^2 - D_{p_3} - D_{\tau_{p_3}}\} = \{0, 3\}$

$\alpha(\ell_1) = \{f_{\ell_1}^1, f_{\ell_1}^2, f_{\ell_1}^3\} = \{L_{\ell_1}, f_{p_1}^1 - L_{\ell_2}, f_{p_1}^2 - L_{\ell_2}\} = \{0, 7, 10\}$

$\alpha(\ell_2) = \{f_{\ell_2}^1, f_{\ell_2}^2, f_{\ell_2}^3\} = \{L_{\ell_2}, f_{p_1}^1 - L_{\ell_1}, f_{p_1}^2 - L_{\ell_1}\} = \{0, 7, 10\}$

$\alpha(\ell_3) = \{f_{\ell_3}^1, f_{\ell_3}^2, f_{\ell_3}^3\} = \{L_{\ell_3}, f_{p_3}^1 - L_{\ell_4}, f_{p_3}^2 - L_{\ell_4}\} = \{0, 5, 8\}$

$\alpha(\ell_4) = \{f_{\ell_4}^1, f_{\ell_4}^2, f_{\ell_4}^3\} = \{L_{\ell_4}, f_{p_3}^1 - L_{\ell_3}, f_{p_3}^2 - L_{\ell_3}\} = \{0, 5, 8\}$

$$\alpha(\ell_5) = \{f^1_{\ell_5}, f^2_{\ell_5}, f^3_{\ell_5}\} = \{L_{\ell_5}, f^1_{p_2} - L_{\ell_6}, f^2_{p_2} - L_{\ell_6}\} = \{0, 0, 3\}$$

$$\alpha(\ell_6) = \{f^1_{\ell_6}, f^2_{\ell_6}, f^3_{\ell_6}\} = \{L_{\ell_6}, f^1_{p_2} - L_{\ell_5}, f^2_{p_2} - L_{\ell_5}\} = \{0, 0, 3\}$$

$$\alpha(\ell_7) = \{f^1_{\ell_7}, f^2_{\ell_7}, f^3_{\ell_7}\} = \{L_{\ell_7}, f^1_{p_4} - L_{\ell_8}, f^2_{p_4} - L_{\ell_8}\} = \{0, 0, 3\}$$

$$\alpha(\ell_8) = \{f^1_{\ell_8}, f^2_{\ell_8}, f^3_{\ell_8}\} = \{L_{\ell_8}, f^1_{p_4} - L_{\ell_7}, f^2_{p_4} - L_{\ell_7}\} = \{0, 0, 3\}$$

We start from the bottom of the decomposition tree. At a leaf node we simply substitute each extreme flow for $f$ in its cost function and obtain the corresponding linear functions of $\lambda$.

At a p-node, we calculate the cost function for each extreme flow, which is the minimum of two piecewise linear and concave functions. Let $C^k_i$ be the cost of sending the $k$th extreme flow through the subgraph represented by node $i$. Figure 4.6 illustrates the cost function of $f^1_{p_1}$. It is the minimum of either (i) sending the lower bound through its right child and the rest through its right child: $C^1_{\ell 1} + C^2_{\ell_2}$ or (ii) the lower bound through its left child and the rest through its right child: $C^1_{\ell 2} + C^2_{\ell_1}$. The breakpoints in this case occur at $\lambda = 0$, $\frac{4}{9}$ and 1. For the interval $[0, \frac{4}{9})$, it is cheaper to send the flower as (ii) while for $[\frac{4}{9}, 1]$ it is cheaper to do as (i). At an s-node, we calculate the cost function for each extreme flow, which is the sum of two piecewise linear and concave functions.

We list the cost functions each node in the decomposition tree as follows:

$$C^1_{\ell_1} = C_{\ell_1}(f^1_{\ell_1}) = C_{\ell_1}(0) = 0$$

$$C^2_{\ell_1} = C_{\ell_1}(f^2_{\ell_1}) = C_{\ell_1}(7) = 19 + 6\lambda$$

$$C^3_{\ell_1} = C_{\ell_1}(f^3_{\ell_1}) = C_{\ell_1}(10) = 25 + 9\lambda$$

$$C^1_{\ell_2} = C_{\ell_2}(f^1_{\ell_1}) = C_{\ell_2}(0) = 0$$

$$C^2_{\ell_2} = C_{\ell_2}(f^2_{\ell_1}) = C_{\ell_2}(7) = 23 - 3\lambda$$

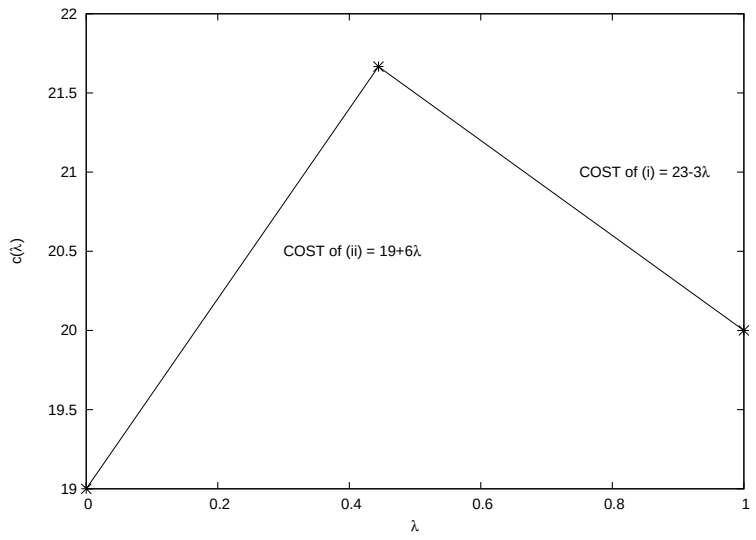



Figure 4.6: *The piecewise and concave cost function for node* $p_1$

$$C^3_{\ell_2} = C_{\ell_2}(f^3_{\ell_2}) = C_{\ell_2}(10) = 29 - 3\lambda$$

$$C^1_{\ell_5} = C_{\ell}(f^1_{\ell_5}) = C_{\ell_5}(0) = 0$$

$$C^2_{\ell_5} = C^1_{\ell_5} = 0$$

$$C^3_{\ell_5} = C_{\ell_5}(f^3_{\ell_5}) = C_{\ell_5}(3) = 14 - 3\lambda$$

$$C^1_{\ell_6} = C_{\ell_6}(f^1_{\ell_6}) = C_{\ell_6}(0) = 0$$

$$C^2_{\ell_6} = C^1_{\ell_6} = 0$$

$$C^3_{\ell_6} = C_{\ell_6}(f^3_{\ell_6}) = C_{\ell_6}(3) = 21 - 14\lambda$$

$$C^1_{\ell_3} = C_{\ell_3}(f^1_{\ell_3}) = C_{\ell_3}(0) = 0$$

$$C^2_{\ell_3} = C_{\ell_3}(f^2_{\ell_3}) = C_{\ell_3}(5) = 7 + 21\lambda$$

$$C^3_{\ell_3} = C_{\ell_3}(f^3_{\ell_3}) = C_{\ell_3}(8) = 10 + 31\lambda$$

$$C^1_{\ell_4} = C_{\ell_4}(f^1_{\ell_4}) = C_{\ell_4}(0) = 0$$

$$C^2_{\ell_4} = C_{\ell_4}(f^2_{\ell_4}) = C_{\ell_4}(5) = 21 - \lambda$$

$$C^3_{\ell_4} = C_{\ell_4}(f^3_{\ell_4}) = C_{\ell_4}(8) = 31 + 2\lambda$$

$$C^1_{\ell_7} = C_{\ell_7}(f^1_{\ell_7}) = C_{\ell_7}(0) = 0$$

$$C^2_{\ell_7} = C_{\ell_7}(f^1_{\ell_7}) = 0$$

$$C^3_{\ell_7} = C_{\ell_7}(f^3_{\ell_7}) = C_{\ell_7}(3) = 17 - 4\lambda$$

$$C^1_{\ell_8} = C_{\ell_8}(f^1_{\ell_8}) = C_{\ell_8}(0) = 0$$

$$C^2_{\ell_8} = C_{\ell_8}(f^1_{\ell_8}) = 0$$

$$C^3_{\ell_8} = C_{\ell_8}(f^3_{\ell_8}) = C_{\ell_8}(3) = 6 + 16\lambda$$

$$C^1_{p_1} = \min(C^1_{\ell 1} + C^2_{\ell_2}, C^1_{\ell 2} + C^2_{\ell_1}) = \min(23 - 3\lambda, 19 + 6\lambda) = \begin{cases} 19 + 6\lambda & \forall\lambda \in [0, \frac{4}{9}) \\ 23 - 3\lambda & \forall\lambda \in [\frac{4}{9}, 1] \end{cases}$$

$$C^2_{p_1} = \min(C^1_{\ell 1} + C^3_{\ell_2}, C^1_{\ell 2} + C^3_{\ell_1}) = \min(29 - 3\lambda, 25 + 9\lambda) = \begin{cases} 25 + 9\lambda & \forall\lambda \in [0, \frac{1}{3}) \\ 29 - 3\lambda & \forall\lambda \in [\frac{1}{3}, 1] \end{cases}$$

$$C^1_{p_2} = \min(C^1_{\ell 5} + C^2_{\ell_6}, C^1_{\ell 6} + C^2_{\ell_5}) = 0 \ \forall\lambda \in [0, 1]$$

$$C^2_{p_2} = \min(C^1_{\ell 5} + C^3_{\ell_6}, C^1_{\ell 6} + C^3_{\ell_5}) = \min(21 - 14\lambda, 14 - 3\lambda) = \begin{cases} 14 - 3\lambda & \forall\lambda \in [0, \frac{7}{11}) \\ 21 - 14\lambda & \forall\lambda \in [\frac{7}{11}, 1] \end{cases}$$

$$C^1_{p_3} = \min(C^1_{\ell 3} + C^2_{\ell_4}, C^1_{\ell 4} + C^2_{\ell_3}) = \min(21 - \lambda, 7 + 21\lambda) = \begin{cases} 7 + 21\lambda & \forall\lambda \in [0, \frac{7}{11}) \\ 21 - \lambda & \forall\lambda \in [\frac{7}{11}, 1] \end{cases}$$

$$C^2_{p_3} = \min(C^1_{\ell 3} + C^3_{\ell_4}, C^1_{\ell 4} + C^3_{\ell_3}) = \min(31 + 2\lambda, 10 + 31\lambda) = \begin{cases} 10 + 31\lambda & \forall\lambda \in [0, \frac{21}{29}) \\ 31 + 2\lambda & \forall\lambda \in [\frac{21}{29}, 1] \end{cases}$$

$$C^1_{p_4} = \min(C^1_{\ell 7} + C^2_{\ell_8}, C^1_{\ell 8} + C^2_{\ell_7}) = 0 \ \forall\lambda \in [0, 1]$$

$$C^2_{p_4} = \min(C^1_{\ell 7} + C^3_{\ell_8}, C^1_{\ell 8} + C^3_{\ell_7}) = \min(6 + 16\lambda, 17 - 4\lambda) = \begin{cases} 6 + 16\lambda & \forall\lambda \in [0, \frac{11}{20}) \\ 17 - 4\lambda & \forall\lambda \in [\frac{11}{20}, 1] \end{cases}$$

$$C^1_{s_1} = C^1_{p_1} + C^1_{p_2} = \begin{cases} 19 + 6\lambda & \forall\lambda \in [0, \frac{4}{9}) \\ 23 - 3\lambda & \forall\lambda \in [\frac{4}{9}, 1] \end{cases}$$

$$C^2_{s_1} = C^2_{p_1} + C^2_{p_2} = \begin{cases} 39 + 6\lambda & \forall\lambda \in [0, \frac{1}{3}) \\ 43 - 6\lambda & \forall\lambda \in [\frac{1}{3}, \frac{7}{11}] \\ 50 - 17\lambda & \forall\lambda \in [\frac{7}{11}, 1] \end{cases}$$

$$C^1_{s_2} = C^1_{p_3} + C^1_{p_4} = \begin{cases} 7 + 21\lambda & \forall\lambda \in [0, \frac{7}{11}) \\ 21 - \lambda & \forall\lambda \in [\frac{7}{11}, 1] \end{cases}$$

$$C^2_{s_2} = C^2_{p_3} + C^2_{p_4} = \begin{cases} 16 + 47\lambda & \forall\lambda \in [0, \frac{11}{20}) \\ 27 + 27\lambda & \forall\lambda \in [\frac{11}{20}, \frac{21}{29}] \\ 48 - 2\lambda & \forall\lambda \in [\frac{21}{29}, 1] \end{cases}$$

$C^1_{p_5} = \min(C^1_{s_1} + C^2_{s_2}, C^1_{s_2} + C^2_{s_1})$ where

$$C^1_{s_1} + C^2_{s_2} = \begin{cases} 35 + 53\lambda & \forall\lambda \in [0, \frac{4}{9}) \\ 39 + 44\lambda & \forall\lambda \in [\frac{4}{9}, \frac{11}{20}] \\ 50 + 24\lambda & \forall\lambda \in [\frac{11}{20}, \frac{21}{29}) \\ 71 - 5\lambda & \forall\lambda \in [\frac{21}{29}, 1] \end{cases}$$

and,

$$C^1_{s_2} + C^2_{s_1} = \begin{cases} 46 + 27\lambda & \forall\lambda \in [0, \frac{1}{3}) \\ 50 + 15\lambda & \forall\lambda \in [\frac{1}{3}, \frac{7}{11}] \\ 71 + 18\lambda & \forall\lambda \in [\frac{7}{11}, 1] \end{cases}$$

So

$$C^1_{p_5} = \begin{cases} \min(46+27\lambda, 35+53\lambda) & \forall\lambda \in [0, \frac{1}{3}) \\ \min(50+15\lambda, 35+53\lambda) & \forall\lambda \in [\frac{1}{3}, \frac{4}{9}] \\ \min(50+15\lambda, 39+44\lambda) & \forall\lambda \in [\frac{4}{9}, \frac{7}{11}] \\ \min(71+18\lambda, 39+44\lambda) & \forall\lambda \in [\frac{7}{11}, \frac{11}{20}] \\ \min(71+18\lambda, 50+24\lambda) & \forall\lambda \in [\frac{11}{20}, \frac{21}{29},] \\ \min(71+18\lambda, 71-5\lambda) & \forall\lambda \in [\frac{21}{29}, 1] \end{cases}$$

$\Longrightarrow$

$$C^1_{p_5} = \begin{cases} 35+53\lambda & \forall\lambda \in [0, \frac{15}{38}) \\ 50+15\lambda & \forall\lambda \in [\frac{15}{38}, \frac{7}{11}) \\ 39+44\lambda & \forall\lambda \in [\frac{7}{11}, \frac{11}{20}] \\ 50+24\lambda & \forall\lambda \in [\frac{11}{20}, \frac{21}{29}] \\ 71-5\lambda & \forall\lambda \in [\frac{21}{29}, 1] \end{cases}$$

#### 4.4.0.3 Construct the Optimal Solution

We now can construct the optimal flow for any given $\lambda \in [0, 1]$. For example, let $\lambda = 0.6$. At $p_5$, the corresponding breakpoint interval is $[\frac{11}{20}, \frac{11}{29})$ so the total cost is $50 + 24\lambda = 50 + 24 \times 0.6 = 64.4$. Furthermore, the minimum cost is obtained at $C^1_{s_1} + C^2_{s_2}$, where $C^1_{s_1} = C^1_{p_1} + C^1_{p_2}$ and $C^2_{s_2} = C^2_{p_3} + C^2_{p_4}$ at $\lambda = 0.6$. The minimum of $C^1_{p_1}$ at $\lambda = 0.6$ is obtained at $f^*_{\ell 1} = 0$ and $f^*_{\ell 2} = 7$. Similarly, we find the optimal flows for the other leaf nodes: $f^*_{\ell 3} = 8$, $f^*_{\ell 7} = 3$ and $f^*_{\ell 4} = f^*_{\ell 8} = 0$.

1. algorithm AL: **begin**
2. Construct the binary decomposition tree for the directed series-parallel graph;
3. From the bottom of the decomposition tree to the top, apply the parametric shortest path procedure. Keep track of which child node gives the minimum between breakpoints. At the same time, compute the lower bound $L_G$ for each node $G$ and its internal demand $D_G$;
4. Construct the optimal flow for any given $\lambda \in [0, 1]$. Start from the root of the decomposition tree to the bottom; at the root node, $f_r^* = L_r^* + D_{\tau_r}$; for each s-node $G$, compute $f_{l_G}^* = f_G^*$ and $f_{r_G}^* = f_G^* - D_{l_G} - D_{\tau_{l_G}}$; for each p-node G, find which breakpoints interval $\lambda$ belongs to, the corresponding child node $i$ and set $f_i^* = f_G^* - L_j$ and $f_j^* = L_j$;
5. Calculate the total flow cost by summing $C_{\ell_1}, ...C_{\ell_A}$;
6. **end** AL;

Figure 4.7: *AL: algorithm for problems with only linear objective functions*

## 4.5 Algorithm for the Case with Only Linear Objective Functions (AL)

In this section, we discuss a special case where the cost functions $C_a(f)$ and $D_a(f)$ are both linear. The single-objective can be written as $[\lambda C_a + (1-\lambda)D_a]f$. Figure 4.7 describes our algorithm. We illustrate the algorithm using the same graph (Figure 4.5) with a new set of cost functions as follows

$\ell_1$: $[3\lambda + 2(1-\lambda)]f$ ; $\ell_2$: $[4\lambda + (1-\lambda)]f$

$\ell_3$: $[2\lambda + (1-\lambda)]f$ ; $\ell_4$: $[3\lambda + (1-\lambda)]f$

$\ell_5$: $[5\lambda + 2(1-\lambda)]f$ ; $\ell_6$: $[4\lambda + 3(1-\lambda)]f$

$\ell_7$: $[6\lambda + (1-\lambda)]f$ ; $\ell_8$: $[4\lambda + 2(1-\lambda)]f$.

From the bottom of the decomposition tree to the top, calculate the lower bounds:

$L_{\ell_1} = L_{\ell_2} = ... = L_{\ell_7} = L_{\ell_8} = 0$

$L_{p_1} = L_{\ell_1} + L_{\ell_2} = 0$; $L_{p_2} = L_{p_3} = L_{p_4} = 0$

$L_{s_1} = \max(0, 0 + 7 + 0) = 7$; $L_{s_2} = \max(0, 0 + 5 + 0) = 5$

$L_{p_5} = L_{s_1} + L_{s_2} = 12$

The flow into $p_5$ has to be at least $f_{p_5} = L_{p_5} + D_{\tau_{p_5}} = 12 + 3 = 15$, which can be satisfied by the available supply.

As proved by Ward (1999), there exists an optimal solution to this problem which is extreme. We mentioned that one of the characteristics of extreme flows is that for each parallel node G in the decomposition tree $\Im$, $f_g > L_g$ for at most one child of G ($r_G$ or $l_G$). In other words, for each subgraph G, $L_G$ is the minimum flow requirement and the cost of sending these flows happens no matter what. What makes the difference is how to send the remaining $f_G - L_{r_G} - L_{l_G}$. The intuition is to use dynamic programming. Start from the bottom of the tree. For each G that is a p-node, find the cheapest child to carry the remaining flow of $f_G - L_{r_G} - L_{l_G}$. Ward shows that if the arc flow costs are linear, for each node G in the decomposition tree, the minimum cost of sending $f$ into G is independent of $f$. Therefore, we only need to solve the parametric shortest path problem on G and send $f_G - L_{r_G} - L_{l_G}$ over this path.

We adopt the algorithm from Raghavan et al. (2002) to solve the parametric shortest path problem. From the bottom of the decomposition tree, for each s-node

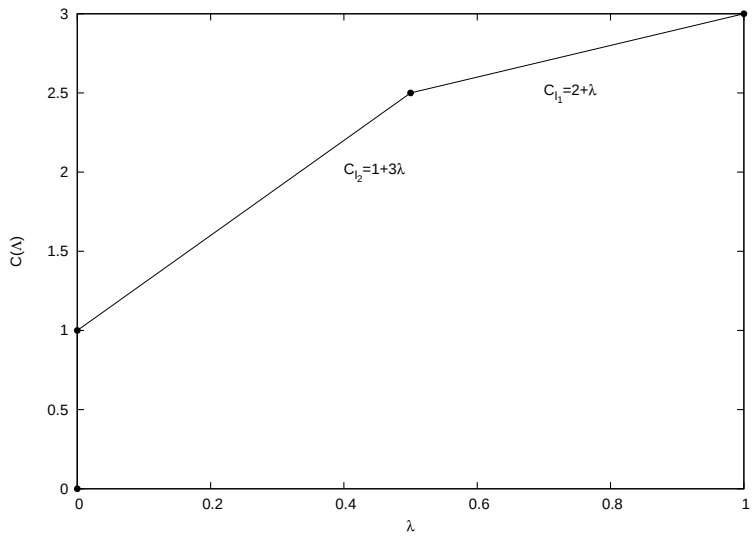



Figure 4.8: *The piecewise and concave cost function for node* $p_1$

we calculate the sum of two piecewise linear and concave functions; for each p-node we look for the minimum of two piecewise linear and concave functions. Along the way, we record the breakpoint(s) for each p-node. For example, Figure 4.8 shows the minimum of the cost functions of $p_1$'s children ($\ell_1$ and $\ell_2$). The breakpoints in this case occur at $\lambda = 0$, 0.5 and 1. For the interval [0,0.5), $\ell_2$ is cheaper while for [0.5,1] $\ell_1$ is cheaper. The cheaper child at each p-node and the corresponding breakpoint intervals in the example are recorded as follows:

$$P_1 \longrightarrow \begin{cases} \ell_2 & \text{if } \lambda = [0, \frac{1}{2}) \\ \ell_1 & \text{if } \lambda = [\frac{1}{2}, 1] \end{cases}$$

$$P_2 \longrightarrow \begin{cases} \ell_5 & \text{if } \lambda = [0, \frac{1}{2}) \\ \ell_6 & \text{if } \lambda = [\frac{1}{2}, 1] \end{cases}$$

$$P_3 \longrightarrow \ell_7 \ \ \forall \lambda = [0,1]$$

$$P_4 \longrightarrow \begin{cases} \ell_7 & \text{if } \lambda = [0, \frac{1}{3}) \\ \ell_8 & \text{if } \lambda = [\frac{1}{3}, 1] \end{cases}$$

$$P_5 \longrightarrow S_2 \ \ \forall \lambda = [0,1]$$

We now can construct an optimal solution for any $\lambda \in [0,1]$. The following steps describe the construction of an optimal solution for $\lambda = 0.4$.

1. Start from the top of the decomposition tree. Since $\lambda = 0.4 \in [\frac{1}{3}, 1]$, for node $p_5$ we choose its cheaper child $s_2$ and set $f^*_{s_1} = L_{s_1} = 7$, $f^*_{s_2} = f^*_{p_5} - L_{s_1} = 15 - 7 = 8$

2. At $s_1$, $f^*_{p_1} = f^*_{s_1} = 7$ and $f^*_{p_2} = f^*_{p_1} - D_{p_1} - D_{\tau_{p_1}} = 7 - 7 = 0$. Similarly, at $s_2$, $f^*_{p_3} = f^*_{s_2} = 8$ and $f^*_{p_4} = f^*_{p_3} - D_{p_3} - D_{\tau_{p_3}} = 8 - 5 = 3$.

3. At $p_1$, since $\lambda = 0.4 \in [0, \frac{1}{2}]$, we choose $\ell_2$. Set $f^*_{\ell_1} = L_{\ell_1} = 0$ and $f^*_{\ell_2} = f^*_{p_1} - L_{\ell_1} = 7 - 0 = 7$. Similarly, at $p_2$, $f^*_{\ell_5} = L_{\ell_6} = 0$; at $p_3$, $f^*_{\ell_3} = f^*_{p_3} - L_{\ell_4} = 8$, $f^*_{\ell_4} = L_{\ell_4} = 0$; at $p_4$, $f^*_{\ell_7} = L_{\ell_7} = 0$, $f^*_{\ell_8} = f^*_{p_4} - L_{\ell_7} = 3$.

The minimum cost is $C_{\ell_1}(f^*_{\ell_1}) + ... + C_{\ell_8}(f^*_{\ell_8}) = 39$.

## 4.6 Algorithm for the Special Case with Only Fixed Objective Functions (AF)

This section deals with another special case where the two objective functions $C_a(f)$ and $D_a(f)$ both have only the fixed-charge. The single-objective transformation is described as follows:

$$\lambda C_a(f) + (1-\lambda)D_a(f) = \begin{cases} 0 & \text{if } f = 0 \\ \lambda k + (1-\lambda)k & \text{if } f > 0 \end{cases}$$

For each node $G$ in the decomposition tree, the cost function tells us that the minimum cost of sending $f$ into G is independent of $f$ as long as $f$ is greater than 0. Observe that in the optimal solutions the flow on each arc is either equal to or greater than its lower bound $L_G$. It follows that the minimum cost of sending $f$ into $G$ is independent of $f$ as long as $f$ is greater than $L_G$, which is itself nonnegative. This observation suggests that for each node in the decomposition tree, we need to consider two possibilities: $f = L_G$ and $f > L_G$. Let $\underline{C}_G = C_G(L_G)$ and $C_G$ be the cost when $f_G \geq L_G$. Figure 4.9 describes our algorithm. We modify the cost functions of the previous example (Figure 4.5) to contain only fixed-charge form and use the resulting problem to illustrate the algorithm. The adjusted cost functions are:

$$\ell_1 = \begin{cases} 0 & \text{if } f = 0 \\ 3\lambda + 2(1-\lambda) & \text{if } f > 0 \end{cases} \quad \ell_2 = \begin{cases} 0 & \text{if } f = 0 \\ 4\lambda + (1-\lambda) & \text{if } f > 0 \end{cases}$$

$$\ell_3 = \begin{cases} 0 & \text{if } f = 0 \\ 2\lambda + (1-\lambda) & \text{if } f > 0 \end{cases} \quad \ell_4 = \begin{cases} 0 & \text{if } f = 0 \\ 3\lambda + (1-\lambda) & \text{if } f > 0 \end{cases}$$

$$\ell_5 = \begin{cases} 0 & \text{if } f = 0 \\ 5\lambda + 2(1-\lambda) & \text{if } f > 0 \end{cases} \quad \ell_6 = \begin{cases} 0 & \text{if } f = 0 \\ 4\lambda + 3(1-\lambda) & \text{if } f > 0 \end{cases}$$

1. algorithm AF: **begin**
2. Construct the binary decomposition tree for the directed series-parallel graph;
3. From the bottom of the decomposition tree to the top, calcuate the lower bounds and at the same time, for leaf node G, $\underline{C_G} = \lambda C_a(L_G) + (1-\lambda)D_a(L_G)$ and $C_G = \lambda k_G + (1-\lambda)k_G$; for s-node G, if $L_G = L_{l_G}$ and $L_G - D_{l_G} - D_{\tau_{l_G}} = L_{r_G}$, $\underline{C_G} = \underline{C_{l_G}} + \underline{C_{r_G}}$; if $L_G = L_{l_G}$ and $L_G - D_{l_G} - D_{\tau_{l_G}} > L_{r_G}$, $\underline{C_G} = \underline{C_{l_G}} + C_{r_G}$; if $L_G > L_{l_G}$ and $L_G - D_{l_G} - D_{\tau_{l_G}} = L_{r_G}$, $\underline{C_G} = C_{l_G} + \underline{C_{r_G}}$; while $C_G = C_{l_G} + C_{r_G}$. for p-node G, $\underline{C_G} = \underline{C_{l_G}} + \underline{C_{r_G}}$ and $C_G = \min(C_{l_G} + \underline{C_{r_G}}, C_{r_G} + \underline{C_{l_G}})$; record the child node $i$ in which we get the minimum between the breakpoints $B_i$.
4. Construct the optimal flow for any given $\lambda \in [0,1]$. Start from the root of the decomposition tree, at the root node, $f_r^* = L_r^* + D_{\tau_r}$; for each s-node $G$, set $f_{l_G}^* = f_G^*$ and $f_{r_G}^* = f_G^* - D_{l_G} - D_{\tau_{l_G}}$; for each p-node G, set $f_i^* = f_G^* - L_j$ and $f_j^* = L_j$, where $i$ is the minimizer we recorded in the previous procedure for the given $\lambda \in [B_i^k, B_i^{k+1}]$.
6. **end** AF;

Figure 4.9: *AF: algorithm for problems with only fixed-charge objective functions*

$$\ell_7 = \begin{cases} 0 & \text{if } f = 0 \\ 6\lambda + (1-\lambda) & \text{if } f > 0 \end{cases} \qquad \ell_8 = \begin{cases} 0 & \text{if } f = 0 \\ 4\lambda + 2(1-\lambda) & \text{if } f > 0 \end{cases}$$

The structure of the decomposition tree remains the same as shown in Figure 4.5(b).

From the bottom of the tree, we calculate the lower bound and cost function for each node as follows:

$$L_{\ell_1} = \underline{C}_{\ell_1} = 0 \text{ and } C_{\ell_1} = 2 + \lambda$$

$$L_{\ell_2} = \underline{C}_{\ell_2} = 0 \text{ and } C_{\ell_2} = 1 + 3\lambda$$

$$L_{\ell_3} = \underline{C}_{\ell_3} = 0 \text{ and } C_{\ell_3} = 1 + \lambda$$

$$L_{\ell_4} = \underline{C}_{\ell_4} = 0 \text{ and } C_{\ell_4} = 1 + 2\lambda$$

$$L_{\ell_5} = \underline{C}_{\ell_5} = 0 \text{ and } C_{\ell_5} = 2 + 3\lambda$$

$$L_{\ell_6} = \underline{C}_{\ell_6} = 0 \text{ and } C_{\ell_6} = 3 + \lambda$$

$$L_{\ell_7} = \underline{C}_{\ell_7} = 0 \text{ and } C_{\ell_7} = 1 + 5\lambda$$

$$L_{\ell_8} = \underline{C}_{\ell_8} = 0 \text{ and } C_{\ell_8} = 2 + 2\lambda$$

$$L_{p_1} = 0, \underline{C_{p_1}} = 0 \text{ and}$$

$$C^1_{p_1} = \min(C_{\ell_1}, C_{\ell_2}) = \min(2 + \lambda, 1 + 3\lambda) = \begin{cases} 1 + 3\lambda & \forall\lambda \in [0, \frac{1}{2}) \\ 2 + \lambda & \forall\lambda \in [\frac{1}{2}, 1] \end{cases}$$

$$L_{p_2} = 0, \underline{C_{p_2}} = 0 \text{ and}$$

$$C^1_{p_2} = \min(2 + 3\lambda, 3 + \lambda) = \begin{cases} 2 + 3\lambda & \forall\lambda \in [0, \frac{1}{2}) \\ 3 + \lambda & \forall\lambda \in [\frac{1}{2}, 1] \end{cases}$$

$$L_{p_3} = 0, \underline{C_{p_3}} = 0 \text{ and}$$

$$C^1_{p_3} = \min(1 + \lambda, 1 + 2\lambda) = 1 + \lambda \;\; \forall\lambda \in [0, 1]$$

$$L_{p_4} = 0,\ \underline{C_{p_4}} = 0 \text{ and}$$

$$C^1_{p_4} = \min(1+5\lambda, 2+2\lambda) = \begin{cases} 1+5\lambda & \forall\lambda \in [0, \frac{1}{3}) \\ 2+2\lambda & \forall\lambda \in [\frac{1}{3}, 1] \end{cases}$$

$$L_{s_1} = 7,$$

$$\underline{C_{s_1}} = C_{p_1} + \underline{C_{p_2}} = \min(2+\lambda, 1+3\lambda) = \begin{cases} 1+3\lambda & \forall\lambda \in [0, \frac{1}{2}) \\ 2+\lambda & \forall\lambda \in [\frac{1}{2}, 1] \end{cases}$$

and

$$C_{s_1} = C_{p_1} + C_{p_2} = \begin{cases} 3+6\lambda & \forall\lambda \in [0, \frac{1}{2}) \\ 5+2\lambda & \forall\lambda \in [\frac{1}{2}, 1] \end{cases}$$

$$Ls_2 = 5,$$

$$\underline{C_{s_2}} = C_{p_3} + \underline{C_{p_4}} = \min(1+\lambda, 1+2\lambda) = 1+\lambda \ \ \forall\lambda \in [0,1]$$

and

$$C_{s_2} = C_{p_3} + C_{p_4} = \begin{cases} 2+6\lambda & \forall\lambda \in [0, \frac{1}{3}) \\ 3+3\lambda & \forall\lambda \in [\frac{1}{3}, 1] \end{cases}$$

$$L_{p_5} = 12,$$

$$\underline{C_{p_5}} = \underline{C_{s_1}} + \underline{C_{s_2}} = \begin{cases} 2+4\lambda & \forall\lambda \in [0, \frac{1}{2}) \\ 3+2\lambda & \forall\lambda \in [\frac{1}{2}, 1] \end{cases}$$

and

$$C_{p_5} = \min(C_{s_1} + \underline{C_{s2}}, C_{s_2} + \underline{C_{s1}}) = \begin{cases} 3+9\lambda & \forall\lambda \in [0, \frac{1}{3}) \\ 4+6\lambda & \forall\lambda \in [\frac{1}{3}, \frac{1}{2}) \\ 5+4\lambda & \forall\lambda \in [\frac{1}{2}, 1] \end{cases}$$

We can calculate the optimal flow and its corresponding cost for any given $\lambda \in [0, 1]$. For example, when $\lambda = 0.4$, the total cost $C_{p_5} = 4 + 6\lambda = 6.4$, which is the sum of $C_{s_2}$ and $\underline{C_{s1}}$. We have $f^*_{s_2} = f^*_{p_5} - L_{s_1} = 15 - 7 = 8$ and $f^*_{s_1} = L_{s_1} = 7$. And consequently $f^*_{p_1} = f^*_{s_1} = 7$, $f^*_{p_2} = f^*_{s_1} - D_{p_2} - D_{\tau_{p2}} = 7 - 7 = 0$, $f^*_{p_3} = f^*_{s_2} = 8$ and $f^*_{p_4} = f^*_{s_2} - D_{p_3} - D_{\tau_{p3}} = 8 - 5 = 3$. Notice that $\underline{C_{s1}} = C_{p_1} + \underline{C_{p_2}}$. We need to consider $C_{p_1}$ and $\underline{C_{p_2}}$. At $\lambda = 0.4$, the former is minimized at $\ell_2$. Consequently $f^*_{\ell_1} = L_{\ell_1} = 0$ and $f^*_{\ell_2} = f^*_{p_1} - L_{\ell_1} = 7 - 0 = 7$. Similarly, we can get $f^*_{\ell_4} = f^*_{\ell_5} = f^*_{\ell_6} = f^*_{\ell_7} = 0$, $f^*_{\ell_3} = 8$, and $f^*_{\ell_8} = 3$.

### 4.6.1 Analysis of the Algorithms

In this section, we analyze the running time of our three algorithms. It needs to construct the binary decomposition tree from a directed series-parallel graph. This construction can be done in O($|A|$), where $|A|$ is the number of leaf nodes (Raghavan et al. (2002).

At a p-node with extreme flow set $\alpha_p = (f^1_p, ...f^k_p)$. The extreme flow sets for its two child nodes are $\alpha_{l_p} = (f^1_p - L_{r_p}, ...f^k_p - L_{r_p}, L_{l_p})$ and $\alpha_{r_p} = (f^1_p - L_{l_p}, ...f^k_p - L_{l_p}, L_{r_p})$, respectively. For each extreme flow $f^i_p$ in $\alpha_p$ we need to compare the cost of either (i) sending $f^i_p - L_{r_p}$ through $l_p$ and $L_{r_p}$ through $r_p$ or (ii) $f^i_p - L_{l_p}$ through $r_p$ and $L_{l_p}$ through $l_p$. There are three steps involved.

1. calculate the sum of $C_{l_p}(f^i_p - L_{r_p})$ and $C_{r_p}(L_{r_p})$

2. calculate the sum of $C_{l_p}(L_{l_p})$ and $C_{r_p}(f^i_p - L_{l_p})$

3. $C_p = \min\{C_{l_p}(f^i_p - L_{r_p}) + C_{r_p}(L_{r_p}), C_{l_p}(L_{l_p}) + C_{r_p}(f^i_p - L_{l_p})\}$

Let $|B^0_{l_p}|$ denote the number of breakpoints for $C_{l_p}(L_{l_p})$ and $|B^i_{l_p}|$ for $C_{l_p}(f^i_p - L_{r_p})$. Similarly, we have $|B^0_{r_p}|$ and $|B^i_{r_p}|$ for node $r_p$. Lemma 3 shows the first step can be done in $O(|B^0_{r_p}| + |B^i_{l_p}|)$ and the second one in $O(|B^0_{l_p}| + |B^i_{r_p}|)$. In addition, the resulting piecewise functions from the first two steps have at most $|B^0_{r_p}|+|B^i_{l_p}|-2$ and $|B^0_{l_p}| + |B^i_{r_p}| - 2$ breakpoints, respectively. From Lemma 4, we know the third step can be done in $O(|B^0_{l_p}|+|B^i_{l_p}|+|B^0_{r_p}|+|B^i_{r_p}|)$ and the resulting piecewise function has at most $|B^0_{l_p}| + |B^i_{l_p}| + |B^0_{r_p}| + |B^i_{r_p}| - 5$ breakpoints.

Similarly, at an s-node we wish to calculate the cost corresponding to sending each flow in its extreme flow set. Let $|B^i_{l_s}|$ be the number of breakpoints for $C_{l_s}(f^i_s)$ and $|B^i_{r_s}|$ for $C_{r_s}(f^i_s - D_s)$, where $f^i_s \in \alpha_s$. The cost of sending $f^i_s$ through $s$ equals to $C_{l_s}(f^i_s)+C_r(f^i_s-D_s)$. Lemma 3 shows this can be done in $O(|B^i_{l_s}|+|B^i_{r_s}|)$, which can be also be written in the same fashion as that for a $p$ node: $O(|B^0_{l_s}| + |B^i_{l_s}| + |B^0_{r_s}| + |B^i_{r_s}|)$, where $|B^0_{l_s}| = |B^0_{r_s}| = 0$. In addition, from lemma 3 we know $C_s(f^i_s)$ has at most $|B^i_{l_s}| + |B^i_{r_s}| - 2$ breakpoints.

Observe that the number of operations at any node $m$ can be written as $\sum_{f^i_m \in \alpha_m} O(|B^0_{l_m}|+|B^i_{l_m}|+|B^0_{r_m}|+|B^i_{r_m}|)$. Therefore, the total number of operations at nodes at depth $d$ on the tree $\Im$ is:

$$\sum_{m \in V^d(\Im)} O(\sum_{f^i_m \in \alpha_m}(|B^0_{l_m}| + |B^i_{l_m}| + |B^0_{r_m}| + |B^i_{r_m}|)) =$$

$$O(\sum_{j \in V^{d+1}(\Im)} \sum_{f^i_j \in \alpha_j}(|B^0_{l_j}| + |B^i_{l_j}| + |B^0_{r_j}| + |B^i_{r_j}|) + \sum_{m \in V^d(\Im),\, m \in \ell(\Im)} \sum_{f^i_m \in \alpha_m}(1)) =$$

$$O(\sum_{j \in V^{d+1}(\Im)} \sum_{f^i_j \in \alpha_j}(|B^0_{l_j}| + |B^i_{l_j}| + |B^0_{r_j}| + |B^i_{r_j}|) + \sum_{m \in V^d(\Im),\, m \in \ell(\Im)}(|\alpha_m|)$$

By recursion, the number of operations at nodes at depth $d$ is bounded by $\mathrm{O}(\sum_{m \in \ell(\Im),\, m \in V^l(\Im)\ for\ l \geq d} |\alpha(m)|)$. In other words the number of operations at nodes

at depth $d$ on the tree is bounded by a constant times the sum of the number of extreme flows of all leaf nodes at depth $d$ or greater. The number of extreme flows for a node at depth $d$ is bounded by $1 + \lceil \frac{d}{2} \rceil$ (Wald (1999)), which is itself bounded by $O(|A|)$. Additionally, the number of leaf nodes is bounded by $O(|A|)$. Overall the operations at depth $d$ is bounded by $O(|A|^2)$. Consequently the number of operations over the whole tree equals to $O(|A|^3)$. Observe that, by the same argument, we know that the number of breakpoints for all nodes at depth $d$ is $O(|A|^2)$ and that the number over the entire tree is $O(|A|^3)$.

We now discuss the construction of the optimal solutions. Suppose we are interested in the set of nondegenerate parametric solutions. In other words, we wish to find a minimal set of Pareto solutions that contain an optimal solution for each $\lambda \in [0,1]$.

In the execution of the algorithm, we keep track of the child that gives the minimum in between breakpoints of each p-node. Thus wee need $\sum_{m \in V(\Im)} O(B_m)$, or $O(|A|^3)$ space over the entire tree.

To obtain the optimal solution for a given $\lambda \in [0,1]$, we traverse the tree from the root to the leaves, following the appropriate extreme flows indicated by the p-node for the the value of $\lambda$. This operation is bounded by the number of nodes in the tree. Thus it takes $O(|A|)$. To obtain the set of all nondegenerate Pareto solutions we need to repeat the above procedure for each breakpoint interval of the root node. Since the number of breakpoints at the root node is bounded by $O(|A|^2)$, the whole procedure takes $O(|A|^3)$.

We now briefly discuss the two special cases. For the first one where the

objective functions have only linear components, we do not need to compute the extreme flow sets, or we can regard it as having only one element in the extreme flow set of each node. For the second case where the objective functions have only fixed cost, we need to consider two possibilities for each node. We can think of it as having only two elements in the extreme flow set for each node. In other words, we can replace $|\alpha_m|$ with a constant in $O(\sum_{j \in V^{d+1}(\Im)} \sum_{f_j^i \in \alpha_j} (|B_{l_j}^0| + |B_{l_j}^i| + |B_{r_j}^0| + |B_{r_j}^i|) + \sum_{m \in V^d(\Im),\, m \in \ell(\Im)} (|\alpha_m|)$. Thus the number of operations and breakpoints at nodes at depth $d$ becomes $\mathrm{O}(\sum_{m \in \ell(\Im),\, m \in V^l(\Im)\ for\ l \geq d}(1))$, or $\mathrm{O}(|A|^2)$ over the entire tree.

## 4.7 Conclusion

Many bicriteria decision-making problems can be modeled as a parametric uncapacitated network design problems on series-parallel graphs. In this chapter, we focused on a subclass of such problems where the objective functions are linear. We adopted the decomposition tree structure described by Valdes et al. (1982) and developed a polynomial algorithm. We also discussed two special cases where we improved the complexity of the algorithm from $\mathrm{O}(|A|^3)$ to $\mathrm{O}(|A|^2)$.

# Chapter 5

# Arc Routing Models for Small Package Local Routing

## 5.1 Introduction

Small package shipping firms rely on daily local delivery and pick-up routes to service their customer base. At the operational level, each service provider (SP) is responsible for a specific delivery area (e.g., territories fall into the same zip code). In practice, an SP is encouraged to follow a master route, which defines a sequence of street segments and the direction in which each street segment is to be traversed for his/her delivery area. Street segments are defined by address ranges. For instance, a street segment may contain building number 1 to 100 on Maple Street. In the road network, a street segment can be either one-way or two-way. On any given day, the exact set of customers to be served along a given street segment may vary.

Servicing the customers in the same order each day (according to a master route of the delivery area) has various advantages for the SPs including gaining familiarity with their service routes and arriving at regular customers at about the same time each day. In addition, this practice improves the efficiency of delivery since packages are loaded into the vehicles in accordance with the master routes. For instance, packages with destinations located on the street segments that appear early in the master route are placed in the front portion of the cargo area where the SP can easily reach them. Our overall objective is to construct efficient master routes

for the service areas. The yearly revenues of the major small package shipping firms in the United States amount to billions of dollars, thus underscoring the economic importance of efficient local service routes.

The issue of planning daily service where the set of customers may vary each day was first recognized by Jaillet (1985, 1988) who proposed the probabilistic traveling salesman problem (PTSP) where each potential customer has a given presence probability on any given day. The problem is to find a master route through all of the nodes that will minimize the total expected (daily) cost of servicing all of the customers.

In the context of small package local operations, the number of possible different street addresses for customer delivery may be really quite large so the PTSP may not be a practical model. It may be more useful to aggregate the set of possible customers into clusters (Campbell (2006)). We propose to partition them into a set of street segments where each segment has a presence probability (probability of requiring service) on a given day. Also, the available historical service data may only be available in terms of street segments not individual delivery points whose individual presence probabilities may be quite small and difficult to estimate.

Given an extended planning horizon (more than one day), if the set of street segments requiring service every day remains unchanged, we only need to solve an uncapacitated arc routing problem once during the entire time horizon. However, in reality, the street segments that need to be visited can vary on a daily basis. Currently, the problem is often approached in a deterministic manner over a single time period (day). More specifically, a (deterministic and single day/period) arc

routing problem (where a designated set of street segments needs to be serviced) is solved. The resulting master route is used over the entire planning horizon. On a particular day, the routes are realized following the pre-designed sequences while skipping the street segments that do not require service. We will refer to this approach as the deterministic arc routing problem (DARP). We mention that DARP belongs to a family of problems known as the mixed rural postman problem (Laporte (1997)). One major problem with this approach, as pointed out by Jaillet (1985), stems from the fact that a good solution when all required street segments are present may not remain a good solution when some street segments are skipped. We use an example in Figure 5.1 to illustrate this point. Given two master routes $t_1 = (s, 1, 3, 4, 5, 6, 2, 7, e)$ and $t_2 = (s, 1, 2, 6, 5, 4, 3, 7, e)$, where $s$ is the starting location and $e$ the ending location, observe that $t_1$ and $t_2$ have the same deterministic length. Now consider on a particular day, street segment 2 does not require a service, the delivery route following the sequence specified in $t_1$ is $r_1 = (s, 1, 3, 4, 5, 6, 7, e)$ while that following $t_2$ is $r_2 = (s, 1, 6, 5, 4, 3, 7, e)$. Clearly, $r_1$ is shorter than $r_2$.

This uncertainty as to whether a street segment requires service on a particular day suggests that it may be beneficial to study the problem in a probabilistic context. One approach models the problem of finding a suitable master route for an extended planning horizon as a probabilistic arc routing problem (PARP) where each street segment has a corresponding presence probability on any given day just as in the PTSP. The next section elaborates upon this problem description.

An alternative approach is to view finding a suitable master route over an extended planning horizon as a multi-period (deterministic) arc routing problem.

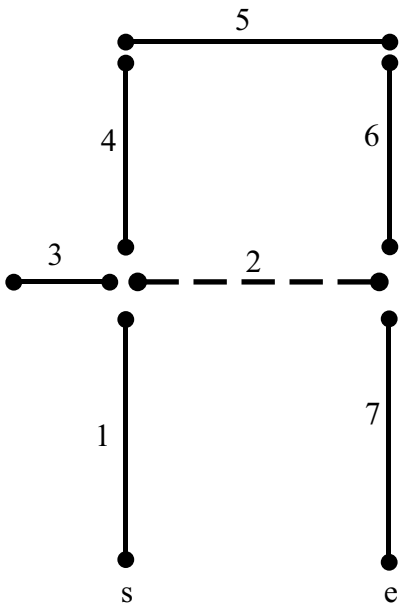



Figure 5.1: *A small example with seven street segments. Street segment 2 (dashed) has a small presence probability. s is the starting location and e the ending location.*

Suppose that we use historical data to create, for a series of days, the sets of street segments that must be serviced for each day. This historical series is a surrogate for the days of the extended planning horizon. Then, we use this historical series as data for a multi-period arc routing problem (MARP). The objective is to find a master route that minimizes the average route length over the given series of days (time periods). The next section discusses this problem definition in more detail.

In the context of the deterministic, probabilistic, and multi-period arc routing models, the remainder of this chapter discusses the advantages and disadvantages of using these three different models for finding master routes in small package local service systems where the customer sets vary from day to day. We also discuss some useful relationships among some of the models.

The next section gives precise descriptions of the three arc routing problems:

the DARP, the PARP, and the MARP. The third section describes a class of examples for the PARP to illustrate how a parameter such as the presence probability of a street segment requiring service can influence the choice of a master route. The following section discusses some local search-based solution approaches for the PARP and MARP. We next describe the implementation and testing of these solution approaches on test problems including some drawn from actual industrial data. The final section summarizes our results and analyzes the use of deterministic vs. probabilistic vs. multi-period arc routing models in small package local routing practice. We also discuss some interesting relationships among the three arc routing models.

Since these master routes are utilized on a daily basis in the small package shipping industry, any minor improvement in the master route length/cost can translate into significant cost savings over a significant time horizon (e.g., one year).

## 5.2 Problem Description of Arc Routing Problems

In this section, we describe three arc routing problems: DARP, PARP, and MARP. All of these arc routing models have the following common inputs: the starting and ending locations (in case the two locations coincide, it becomes the depot), a set of arcs (street segments), the length of each street segment, the length of the shortest path between an endpoint of any segment to an endpoint of any other segment, and the length of the shortest path to/from the ending/starting location from/to the endpoint of any street segment.

### 5.2.1 Deterministic Arc Routing Problem

The DARP has the following description. Given the common inputs, and a set of street segments (arcs) that must be serviced (traversed), find the master route, which starts at the starting location, traverses all the arcs and returns to the ending location, which has the minimum length. DARP belongs to a well-known class of arc routing problems known as the mixed rural postman problem (MRPP). Many solution approaches for this class of problems rely on transforming it into traveling salesman problems (Laporte (1997)). Comprehensive surveys on MRPP as well as the arc routing problem in general can be found in Eiselt et al. (1995), Assad and Golden (1995). In addition, Corberan et al. (2000) present an approximate algorithm based on the resolution of some flow and matching problems as well as a tabu search implementation heuristic approaches to solve MARP.

Note that once we have specified the sequence order for visiting the arcs that must be traversed and the direction in which these arcs are traversed, then it is straightforward to calculate the total route length. After traversing one arc, we always use the shortest path from the endpoint of the just traversed arc to the start of the next arc to be traversed. Since these shortest path lengths and the street segment lengths are part of the common input for the arc routing models, the total route length can be easily computed.

### 5.2.2 Probabilistic Arc Routing Problem

The PARP has the following description. Given common inputs, a set of street segments (arcs) that must be serviced (traversed), and the presence probabilities (probabilities of each segment requiring a visit on a particular day), find the master route, which starts from the starting location, traverses all the arcs and returns to the ending location, which has the minimum expected length.

We now discuss the calculation of the expected length. Our expression is derived from Bertsimas et al. (1990). Without loss of generality, we consider the master route $t = (s, 1, 2, ..., n, e)$, where $s$ is the starting location and $e$ the ending location. Given the presence probability $p_i$ (probability that street segment (arc) $i$ requires a visit on a particular day), we define $q_i = (1 - p_i)$ as the probability that arc $i$ does not require a visit. We use $i_0$ and $i_1$ to represent the entry point and the exit point of $i$, whose length is represented by $d(i_0, i_1)$. Also, let $d(i_1, j_0)$ be the shortest path from street segment $i$ to street segment $j$ on the street network. The expected length of $t$ can be computed with the following expression:

$$E[L(\tau)] = \sum_{i=1}^{n} d(s, i_0) p_i \prod_{k=1}^{i-1} q_k + \tag{5.1}$$

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} d(i_1, j_0) p_i p_j \prod_{k=i+1}^{j-1} q_k + \tag{5.2}$$

$$\sum_{i=1}^{n} d(i_1, e) p_i \prod_{k=i+1}^{n} q_k + \tag{5.3}$$

$$\sum_{i=1}^{n} d(i_0, i_1) p_i. \tag{5.4}$$

The first component of the equation is the expected cost of using the path from the starting location to a street segment $i$ while the third component is that of using the

path from a street segment $i$ back to the ending location. The second component is the expected cost associated with using the path between street segments $i$ and $j$. The last component is the expected cost of using some street segment. The expected cost of the path is based on the probability that the street segments at both ends of the path are realized, the probability that none of the street segments in between are realized, and the length of the path (i.e., the distance from the exit point of the starting segment to the entry point of the ending segment). The expected cost of using a street segment depends on the probability of it being realized (presence probability) and its distance (length from the entry point to the exit point of the street segment). Appendix D provides a small example to illustrate the expected length calculation.

To the best of our knowledge, there is no literature devoted to the PARP. As indicated previously, a closely related problem with presence probabilities is the probabilistic traveling salesman problem (PTSP), which has been studied in the existing literature. The PTSP was first introduced by Jaillet (1988). Laporte, Louveaux, and Mercure (1994) formulated the PTSP as a stochastic integer programming problem and proposed an exact algorithm based on an integer L-shaped method. Additionally, several heuristics have been studied by Bertsimas and Howell (1993), Campbell (2005), Campbell and Thomas (2006), and Liu (2007).

### 5.2.3 Multi-Period Arc Routing Problem

The MARP can be stated as follows. Given the common inputs, a set of time periods (days), and for each day a set of street segments (arcs) that must be serviced on that day, find the master route, which begins from the starting location, traverses all the arcs and returns to the ending location, which has the minimum average length over all of the time periods (days).

To the best of our knowledge, there is no literature devoted to the MARP. However, the essence of MARP can be viewed as an optimization-simulation approach, specifically a sample path optimization, where each time period represents a simulation of the PARP and we use all of the simulations to obtain an optimization of the PARP. Fu (2002) provides an excellent survey on the topic of integrating optimization techniques and simulation practice in general. For the specific sample average approximation method, Verweij et al. (2003), Kleywegt et al. (2002), Shapiro and Nemirovski (2005), Linderoth et al. (2006), Shapiro (2007) examine it in the setting of two-stage discrete stochastic optimization models. Additionally, Levi et al. (2006) study this type of technique in the context of inventory models where they consider the single-period and the multi-period newsvendor problems.

## 5.3 Example of PARP Characteristics

The description of the PARP is straightforward, but understanding the difference between a deterministic arc covering problem and the PARP is more subtle. It is sometimes difficult to see how the probability that an arc requires services on

a particular day can affect the configuration of the optimal route. In order to illustrate some of these concepts, consider the following example. Let $i_1, i_2, ..., i_n$ and $j_1, j_2, ..., j_n$ be the $2n$ nodes in the problem. The arcs are:

1. $(i_k, i_{k+1})$ $k = 1, ...(n-1)$
2. $(i_n, i_1)$
3. $(j_k, j_{k+1})$ $k = 1, ..., (n-1)$
4. $(j_n, j_1)$
5. $(i_k, j_k)$ $k = 1, ..., n$
6. $(j_k, i_{k-1})$ $k = 2, ..., n$
7. $(j_1, i_n)$

All arcs lengths are one. The arcs that must be traversed are:

1. Sets 1 and 2: $(i_k, i_{k+1})$ $k = 1, ..., (n-1)$ and $(i_n, i_1)$; all arcs in the outer ring must be traversed with presence probability one;
2. Sets 3 and 4: $(j_k, j_{k+1})$ $k = 1, ..., (n-1)$ and $(j_n, j_1)$; all arcs in the inner ring must be traversed with presence probability $p$;
3. all other arcs do not need to be traversed. See Figure 5.2 for a depiction of the example.

Now consider two possible covering strategies:

Strategy 1: traverse all arcs in the outer ring and then traverse each arc in the inner ring.

Strategy 2: alternate between traversing an arc in the outer ring and (if necessary) an arc in the inner ring; $(..., (i_{k-1}, i_k), (i_k, j_k), (j_k, j_{k+1}), (j_{k+1}, i_k), ...)$. See Figure 5.3 for an example.

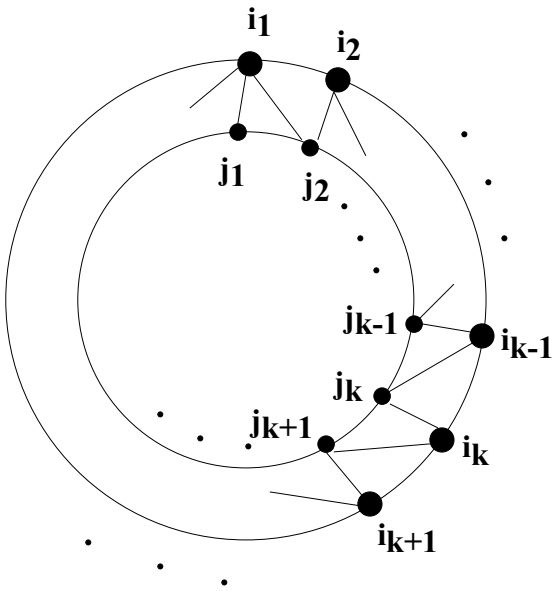



Figure 5.2: *Probabilistic Arc Covering Example*

Both strategies terminate by returning to the original starting node (depot). The expected length of Strategy 1 is $n + 1 + n + 1 = 2n + 2$, independent of the value of $p$. The expected length of Strategy 2 is $n + [n * p * 3] = n(3p + 1)$.

If $p > 1/3$ and $n$ sufficiently large, then Strategy 1 dominates. Otherwise, Strategy 2 dominates. As $p$ approaches 1, the expected length of Strategy 2 is approximately twice the expected length of Strategy 2. As $p$ approaches 0, the expected length of Strategy 1 is approximately twice the expected of Strategy 2. So

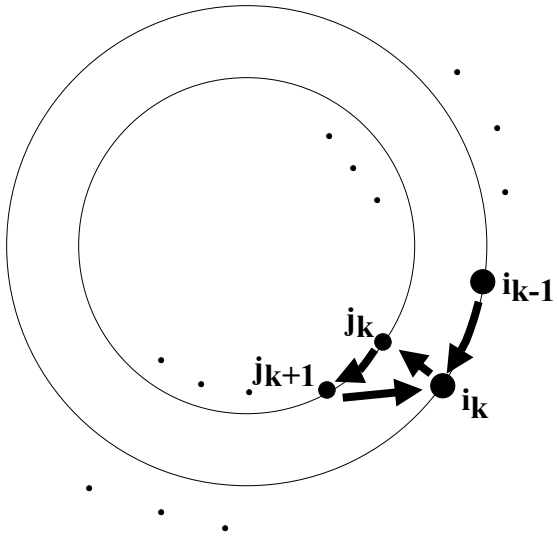



Figure 5.3: *Probabilistic Arc Covering Strategy 2*

the value of the presence probability $p$ plays a significant role in determining the optimal probabilistic arc covering strategy.

The intuition behind strategy 1 is that if the probability $p$ is sufficiently large (i.e., $p > 1/3$), then it is worthwhile to traverse all the arcs on the outer circle and then move onto the inner circle to traverse all of the arcs there. However, if the probability $p$ is sufficiently small (i.e., $p < 1/3$), then the occurrence of arcs on the inner circle is sufficiently small that it is worthwhile to have the master route stay on the outer ring and to make a special trip to the inner ring each time that there is an arc to be traversed and then return to the outer ring.

## 5.4 Suitability of the Arc Routing Models in Small Package Local Service Operations

As discussed in Section 5.1, arc routing models are well suited to model the large number of customer locations (about 1000 on average for the five service routes used in our computational experiments) that can require service over a significant planning horizon (e.g., for a small package local routing system over a 30 day planning horizon). Instead of having a very large number of customer nodes in the PTSP, we represent the customer locations as a moderate number of street segments to be covered. Finding the optimal solution to a PTSP with hundreds or thousands of nodes is rather challenging with currently available solution techniques (see Campbell and Thomas (2008)). In our computational results for the arc routing models, we will see that one of our suggested solution procedures can effectively handle the arc routing model derived from a 30-day model. In addition, as noted in Section 5.1, it may be much easier to estimate presence probabilities for street segments instead of individual addresses whose individual service frequencies may be quite small, even when the corresponding street segment service frequency is relatively large.

Another issue is the suitability of using arc routing models for small package local service operations. Suppose that an arc representing 1 to 100 Maple Street must be traversed. Assume that there are actually three customer locations, at 1, 38, and 70 Maple Street that must be serviced. The description of all three arc routing models states that the entire arc or street segment must be traversed. In reality, the

three customer locations on Maple Street will be covered if the sub-segment from 1-70 Maple Street is traversed. Thus, the arc routing model may overestimate the mileage and route that must be used to cover the customer locations. However, this overestimation should not affect our analysis in determining the relative suitability of the three arc routing models in evaluating and identifying the preferred master route. If we wish to use an arc routing model in an operational context for a small package shipping firm, it is probably necessary to relax the constraint that the entire street segment must always be covered in the arc routing model.

Another issue is that some customer locations or their corresponding street segments may have implicit time windows associated with them (see Wong (2008)). Customer locations may correspond to commercial stores whose operations have evolved to expect their small package deliveries at around the same time each day. The most common type of time window is when the customer deliveries are expected to be before lunchtime (say 1:00 PM). We can address this issue by partitioning the set of arcs to be covered into two subsets, the arcs corresponding to customer locations that normally expect to be serviced before lunch or after lunch. Then we can solve two separate routing problems (corresponding to the two subsets) and then combine the two routes into a solution for the entire arc routing problem. Note that in order to utilize such a partitioning strategy, a great deal of information is required about the customer service history, or expected delivery time is required.

## 5.5 Solution Procedures for Arc Routing Models

The previous sections introduced DARP, PARP, and MARP and discussed characteristics of these models. For our computational tests, the major small package shipper that we worked with provided a sophisticated state-of-the-art procedure for solving the DARP. For the remaining two arc routing models, we utilized local-search based solution techniques that we now describe in detail.

### 5.5.1 Probabilistic Local Search Procedure

For the PARP, we used a solution procedure that adapted local search approaches to our probabilistic context. Our solution heuristic incorporates the presence probabilities using two local search procedures, namely 1-p-Shift and 2-p-Opt. They act as local improvement techniques for the current solution method, which is essentially an efficient TSP heuristic based on the Lin-Kernighan (Helsgaun, 2000) algorithm. We now describe 1-p-Shift and 2-p-Opt in the context of arc routing.

Bertsimas and Howell (1993) provide a clear description of 1-p-Shift and 2-p-Opt, which are designed primarily for the PTSP problem. In the PARP, we consider street segments (arcs) instead of nodes. Accordingly, the local search needs to be adjusted. In particular, we need to decide on the directions (i.e., from which end point to enter) of the street segments. Consider the starting route given in Figure 5.4 where the dashed arcs represent the arcs that must be covered and arc $h$ has nodes $h_0$ and $h_1$ as its end points. The solid directed arcs between a pair of dashed arcs represent the path taken between the two dashed arcs. For the sake of simplicity,

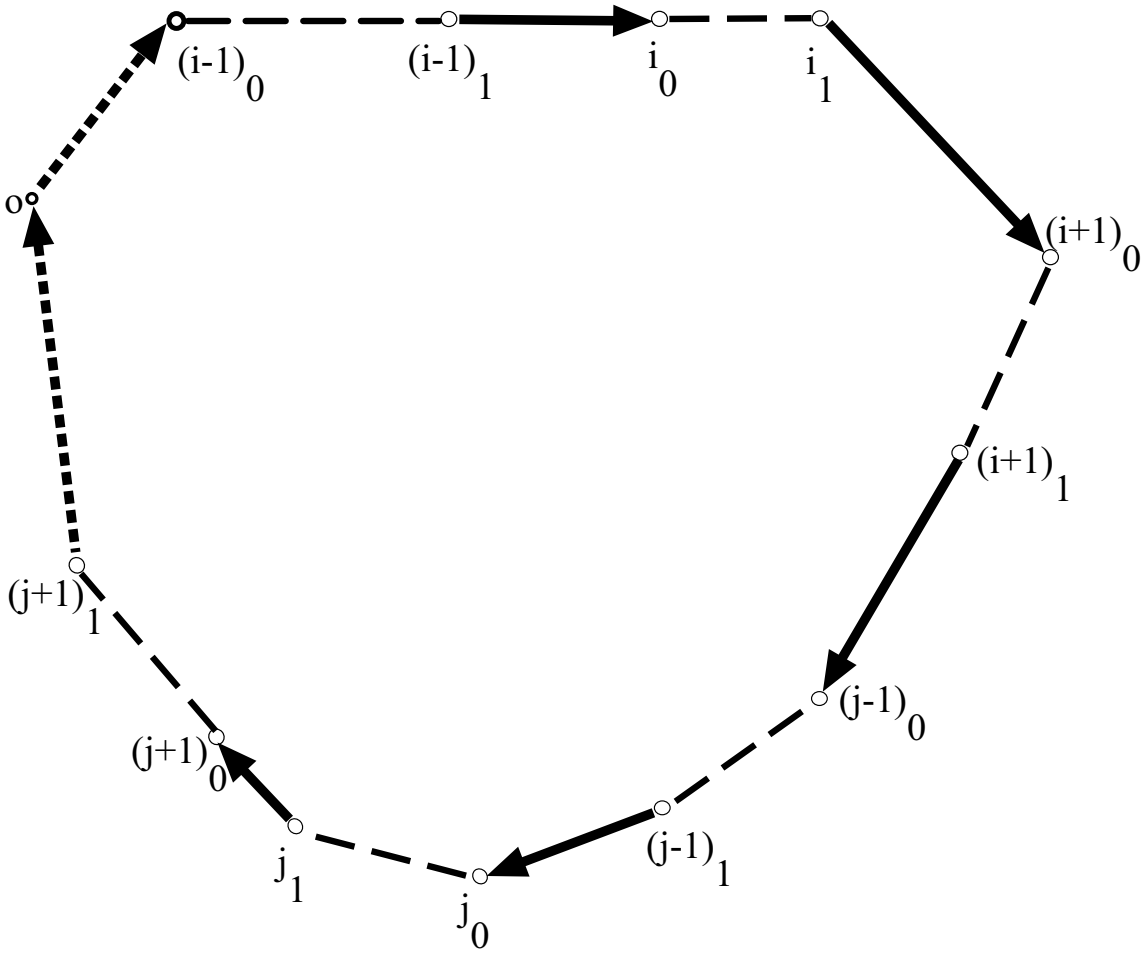



Figure 5.4: *The original tour. The dashed arcs represent the arcs that must be covered and arc h has nodes $h_0$ and $h_1$ as its end points. The solid directed arcs between a pair of dashed arcs represent the path taken between the two dashed arcs. The two dotted arcs represent the path from the depot (node 0) to an arc that must be traversed and the path from an arc that must be traversed to the depot.*

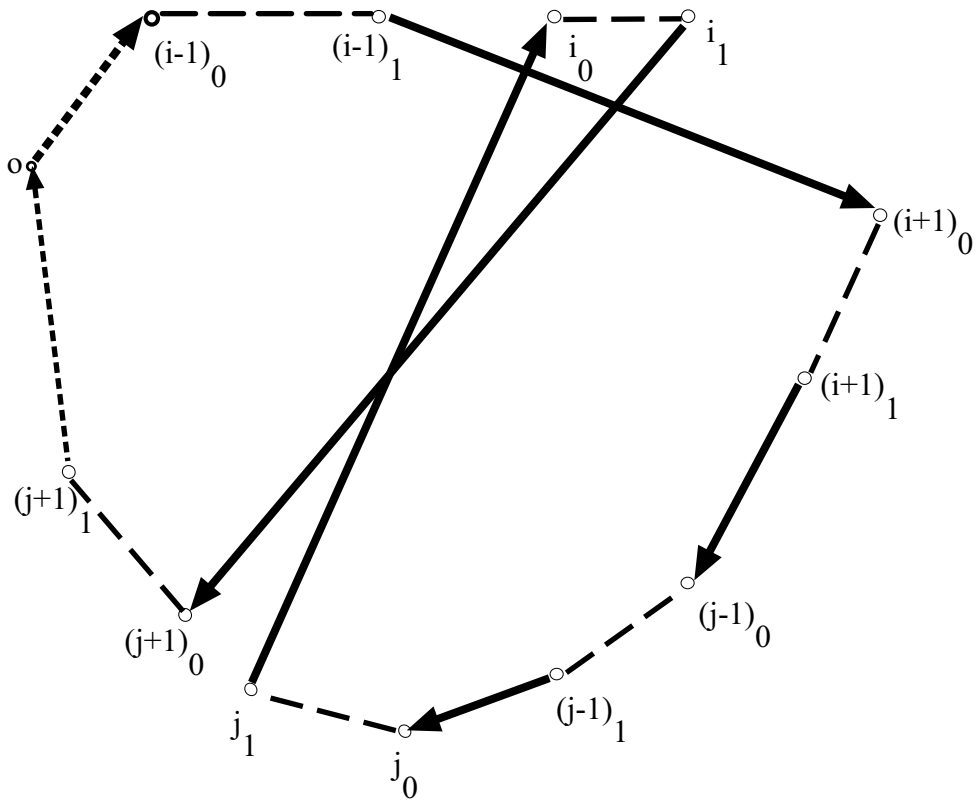



Figure 5.5: *Shift arc i to position j: direction of i remains unchanged*

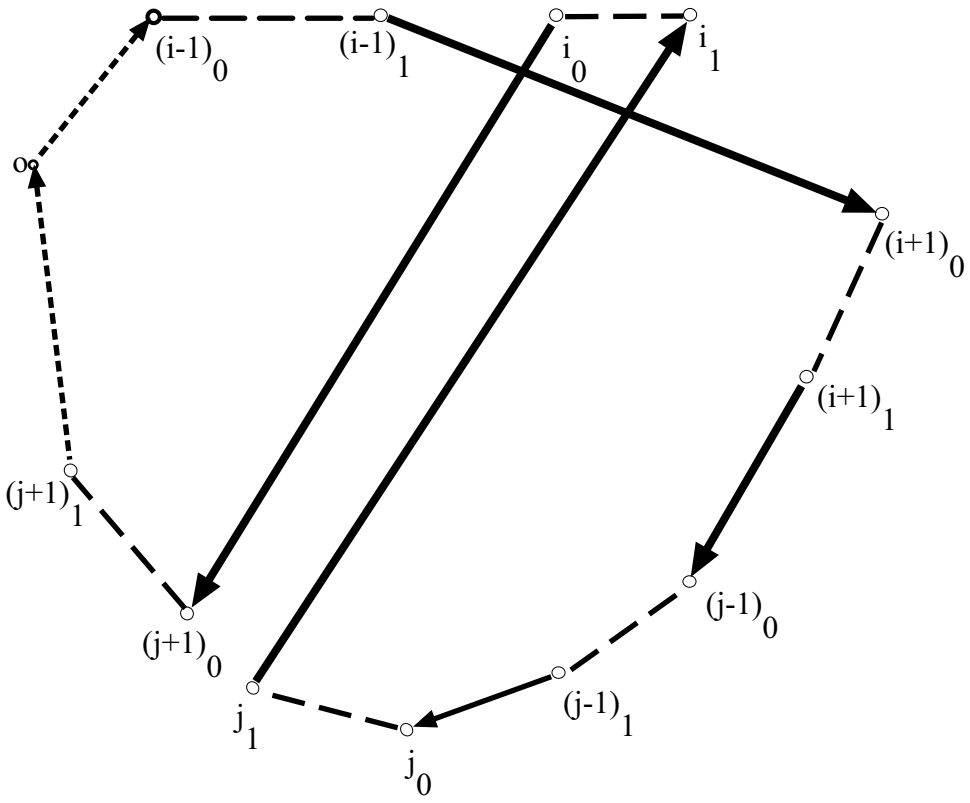



Figure 5.6: *Shift arc i to position j: direction of i is switched*

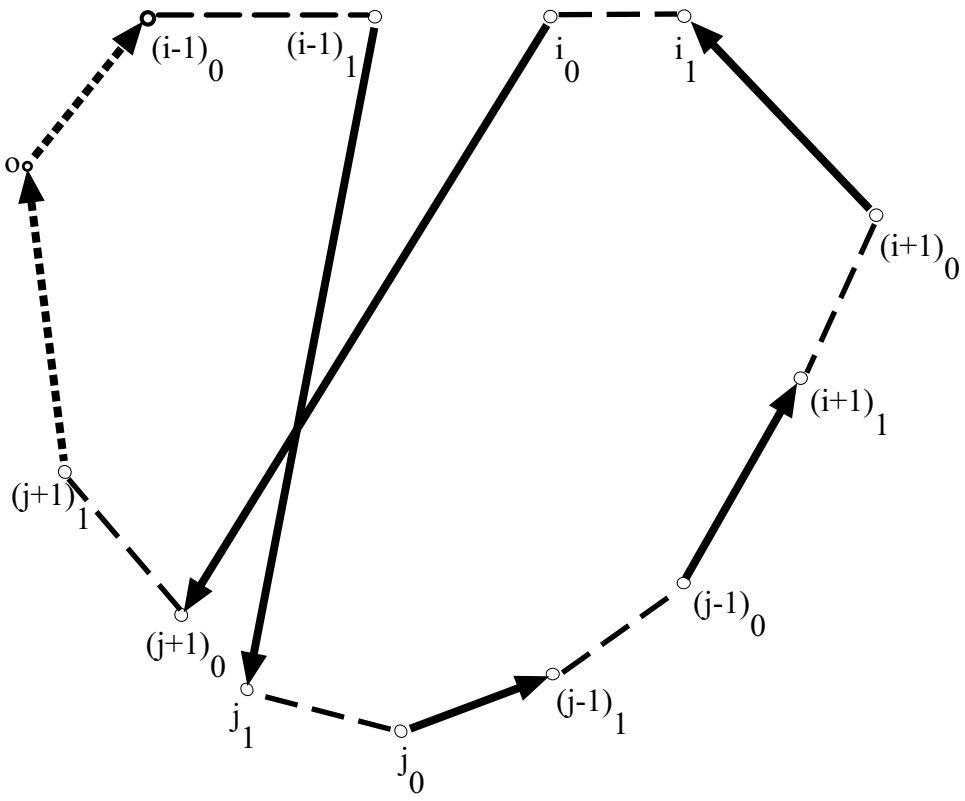



Figure 5.7: *An example of 2-p-Opt: reverse the direction of arcs between i-1 to j+1*

assume the route starts and ends at the same location: the depot (node 0). The two dotted arcs represent the path from the depot to an arc that must be traversed and the path from an arc that must be traversed to the depot. Note that once an ordering of all the arcs that must be traversed and the direction in which each such (dashed) arc is traversed, the route is completely specified. That is, the remaining arcs (the solid and dotted arcs) are completely specified.

Figures 5.5 and 5.6 illustrate two possible options to implement the 1-p-Shift and involve moving arc $i$ from its position on the starting route into the position between arc $j$ and arc $(j+1)$. Figure 5.5 represents the situation when arc $i$ is traversed in the same direction as it was in the starting route of Figure 5.4. Figure 5.6 represents the situation when arc $i$ is traversed in the opposite direction as it was in the starting route. The 2-p-Opt for PARP is more straightforward and similar to the classical 2-Opt procedure where two arcs are removed and two new arcs are inserted in order to form a new route. Consider its implementation on arcs $(i-1)$, $i$, $j$, and $(j+1)$. We simply remove the two arcs connecting arcs $(i-1)$ and $i$ and arcs $j$ and $(j+1)$, reverse the directions of all the arcs in between $(i-1)$ and $(j+1)$ and insert two new arcs as illustrated in Figure 5.7.

Note that for the special case of $i = j$, the 2-p-Opt is equivalent to reversing the direction in which arc $i$ traversed.

Each time a 1-p-Shift or 2-p-Opt movement is proposed, the algorithm calculates the change in the expected length. If there is an improvement in the expected length of the solution, then the movement is accepted; otherwise, the movement is rejected.

**Probabilistic Local Search Solution Procedure**

- 0) Input: a list of street segments (arcs) that requires service; presence probabilities for each street segment on the list, an initial master route stating how to traverse the list of street segments, and an iteration limit. Evaluate the expected length of the initial master route. Initialize the total number of iterations to zero;
- 1) Apply a probabilistic 1-Opt or 2-Opt improvement technique to the current master route. Evaluate the expected length of the new master route. If the expected length of the new master route is less than that of the current master route, the new master route becomes the current master route;
- 2) Increase the total number of iterations by one. If the total number of iterations is less than the iteration limit, go back to Step 1. Otherwise, STOP.

Each proposed solution change of Step 1 requires that the expected cost of the new route must be calculated. We identified various techniques to reuse the computation of the expected value for the current solution in order to reduce the computation time. However, in our implementation of the probabilistic local search, we did not utilize such computational reduction techniques. Even with these techniques, the computational burden of these expected value computations is quite substantial. For the PTSP, various researchers (for example, see the survey by Campbell and Thomas (2008)) have noted the substantial burden in computing expected value computations for large-scale problems greater than about 100 nodes even with some proposed possible remedies. We would expect similar results for the

PARP and the computational results of the next section confirm this belief. We will also see that solving comparably sized MARPs can be significantly easier.

### 5.5.2 The Multi-Period Evaluation Procedure

As we have seen previously, finding a master route for a set of street segments over an extended planning horizon can be cast as a multi-period arc routing problem where the objective is to find the master route that minimizes the average length of the route over the planning period. A key issue is to generate a series of street segment sets corresponding to the set of arcs that must be serviced on each day. As long as this generated series approximates the service requirements over the extended planning horizon, we can expect the master route selected by the MARP to be a reasonably good solution. One possible way of doing this task is to utilize the historical data concerning the set of customer locations that the SP had to visit on a particular day. Then, we could derive the set of arcs requiring service on the particular day. This type of historical data is often available in small package shipping firms.

Observe that MARP has two advantages over PARP. First, evaluating the objective function value for a given master route is much simpler for the MARP than the PARP. The PARP requires a relatively expensive expected value computation of the given master route. For each time period, the MARP must evaluate the (deterministic) route length of the given master route. So, if the number of time periods is moderate (e.g., in our test computations we used 20 or 30 as the number

of time periods), the computational requirements should be reasonable. Second, by evaluating the objective function over historical data, MARP incorporates real-world correlations among the street segment presence probabilities while the expected length formulation used in PARP assumes such probabilities to be independent.

Notice that when the MARP objective function can be efficiently computed, we can employ the basic one-shift and 2-Opt movements as described in the previous subsection as a local improvement heuristic. We can summarize this solution approach in the following way.

**Multi-period Solution Procedure**

- 0) Input: for each one of $n$ time periods, a list of street segments (arcs) that requires service; a summary list of street segments that is the union of the list of street segments requesting service for each time period; an initial master route stating how to traverse the summary list of street segments; an iteration limit. Evaluate the average route length of the master route over the $n$ day time period; Initialize the total number of iterations to zero.

- 1) Apply a 1-Opt or 2-Opt improvement technique to the current master route; evaluate the new master route over the $n$ time periods. If the average route length (over the $n$ time periods) of the new master route is less than that of the current master route, the new master route becomes the current master route;

- 2) Increase the total number of iterations by one. If the total number of iterations is less than the iteration limit, go back to Step 1. Otherwise, STOP.

## 5.6 Computational Results

We implemented the solution procedures for the PARP and MARP described in the previous section in VC++ 6.0 and tested them on a computer with Pentium IV 2GHz and 1.24GB RAM. As discussed previously, the major small package shipping firm that we worked with supplied a sophisticated state-of-the-art procedure for obtaining solutions to the DARP. This section discusses the results of the computational tests with these solution procedures and their implications concerning the utility of these three arc routing models for small package shipping firm local operations. The next subsection describes two sets of test problems, one is drawn from actual industrial data provided by the major small package shipping firm while the other is computer-generated, used for evaluating the performance of these two solution techniques. The second subsection describes the results of the computational tests and some implications of these results.

### 5.6.1 Generating Test Problems

We first describe the set of test problems drawn from actual industrial data. We collected data on five local service routes used by a major small package shipping firm. Each local service route encompasses both commercial and residential areas and required a single service provider to handle the daily work. The service routes were located in three different states. The major small package shipping firm provided the street segments and their lengths as well as the shortest path length between any two endpoints of any two street segments and between the start-

| | Number of Days in Historical Study Interval | Number of Unique Street Segments Served During Historical Study Interval |
|---|---|---|
| Service Route 1 | 30 | 235 |
| Service Route 2 | 30 | 228 |
| Service Route 3 | 20 | 226 |
| Service Route 4 | 30 | 169 |
| Service Route 5 | 30 | 147 |

Table 5.1: *Service Route Characteristics. The number of street segments include the starting and ending locations.*

ing/ending location and the endpoint of any street segment. All of these lengths are derived from the underlying street network of the service territory associated with each route. For each local service route, we also collected daily customer demand data over a historical study interval consisting of 20-30 days. Table 5.1 gives some summary statistics for these service routes.

For each local service route, we derived a corresponding test problem, referred to as Service Route 1 to 5. The list of street segments that must be traversed corresponded to the list of unique street segments serviced during the historical study interval. The presence probability for a street segment is the ratio of the number of days during the historical study interval when the street segment had at least one customer demand to the number of days in the historical study interval. For the number of time periods and the series of street segment sets corresponding to the set of street segments that must be traversed during each time period, we used the number of days in the historical study interval and the daily set of streets segments that must be traversed.

Next, we create the set of computer-generated test problems, referred to as

Problem Set 1 to 5. Each problem has 200 street segments (including the starting and ending locations) and a 30-day study interval. The street segments are randomly placed on a $50 \times 50$ square grid. The coordinates for the starting and ending locations are $(24.5, 0)$ and $(25.5, 0)$, respectively. Euclidean distances are used as the lengths of the street segments as well as the shortest path length between any two endpoints of any two street segments and between the starting/ending location and the endpoint of any street segment. Each street segment presence probability is randomly selected from a uniform distribution on the interval $(0, 1]$. The daily realized street segment data is generated according to the presence probabilities. For example, if the presence probability for a street segment is 0.5 then we randomly select 15 $(= 30 \times 0.5)$ days and create a service request for this segment on each of these six days. We provide the data files for Problem Set 1 in Appendix C. (Given the length of this dissertation, we do not include data files for all the problems. Interested readers may contact the author [1] for the complete data set.)

### 5.6.2 Empirical Evaluation of Master Routes Produced by the Three Arc Routing Models

We used the ten test problems described in the previous subsection to perform various types of evaluations and comparisons. For each test problem, we obtained three master routes (one master route solution obtained by solving each of the three arc routing models). We obtained the solutions by using the solution procedures de-

[1] sichen80@gmail.com

| | Multi-period Evaluation Procedure CPU Time (secs) | Probabilistic Local Search Procedure CPU time (secs) |
|---|---|---|
| Service Route 1 | 15 | 13553 |
| Service Route 2 | 13 | 11818 |
| Service Route 3 | 9 | 11482 |
| Service Route 4 | 5 | 2761 |
| Service Route 5 | 3 | 1567 |

Table 5.2: *Problem Sets Drawn From Actual Industrial Data: Computing Time for Proposed Solution Procedures.*

| | Multi-period Evaluation Procedure CPU Time (secs) | Probabilistic Local Search Procedure CPU time (secs) |
|---|---|---|
| Problem Set 1 | 6 | 5861 |
| Problem Set 2 | 6 | 6058 |
| Problem Set 3 | 6 | 5690 |
| Problem Set 4 | 6 | 5786 |
| Problem Set 5 | 6 | 6008 |

Table 5.3: *Computer Generated Problem Sets: Computing Time for Proposed Solution Procedures.*

scribed in the previous section. The computation times for the multi-day evaluation (MARP) solution procedure and the probabilistic local search (PARP) solution procedure are given in Tables 5.2 and 5.3. Then we performed three types of evaluations using these three arc routing model solutions.

First, we evaluated the three solutions using the total route length criteria of the DARP. Notice that, due to proprietary considerations, we use a normalized cost instead of real mileage. As expected, the DARP master route was the best in terms of the total route length criteria. See Tables 5.4 and 5.5 for this evaluation.

Next, we evaluated the three solutions using the average route length criteria

| | Standard Deterministic Solution (normalized cost) | Multi-day Evaluation Solution (normalized cost) | Probabilistic Location Search Solution (normalized cost) |
|---|---|---|---|
| Service Route 1 | 1.0000 | 1.1206 | 1.1042 |
| Service Route 2 | 1.0000 | 1.2045 | 1.1873 |
| Service Route 3 | 1.0000 | 1.0869 | 1.1611 |
| Service Route 4 | 1.0000 | 1.0585 | 1.0852 |
| Service Route 5 | 1.0000 | 1.1448 | 1.1427 |

Table 5.4: *Problem Sets Drawn From Actual Industrial Data: Master Route Quality using the DARP objective function.*

| | Standard Deterministic Solution | Multi-day Evaluation Solution | Probabilistic Location Search Solution |
|---|---|---|---|
| Problem Set 1 | 16.8684 | 20.2701 | 19.0179 |
| Problem Set 2 | 17.9758 | 21.8542 | 21.0603 |
| Problem Set 3 | 16.3456 | 20.3726 | 20.2638 |
| Problem Set 4 | 17.2726 | 19.2191 | 19.3256 |
| Problem Set 5 | 18.0083 | 20.5402 | 20.6516 |

Table 5.5: *Computer Generated Problem Sets: Master Route Quality using the DARP objective function.*

| | DARP Solution (normalized length) | MARP Solution (normalized length) | PARP Solution (normalized length) |
|---|---|---|---|
| Service Route 1 | 1.0000 | 0.9841 | 0.9783 |
| Service Route 2 | 1.0000 | 0.9522 | 0.9502 |
| Service Route 3 | 1.0000 | 0.9817 | 0.9768 |
| Service Route 4 | 1.0000 | 0.9888 | 0.9883 |
| Service Route 5 | 1.0000 | 0.9885 | 0.9831 |

Table 5.6: *Problem Sets Drawn From Actual Industrial Data: Master Route Quality using the PARP Objective Function.*

| | DARP Solution | MARP Solution | PARP Solution |
|---|---|---|---|
| Problem Set 1 | 11.3491 | 11.0164 | 10.9051 |
| Problem Set 2 | 11.8528 | 11.4514 | 11.4345 |
| Problem Set 3 | 11.3374 | 10.896 | 10.8406 |
| Problem Set 4 | 11.3894 | 10.9851 | 10.953 |
| Problem Set 5 | 11.9243 | 11.5659 | 11.5987 |

Table 5.7: *Computer Generated Problem Sets: Master Route Quality using the PARP Objective Function.*

of the MARP. Tables 5.8 and 5.9 show that the MARP and PARP solutions are better than the DARP solution using the average length criteria. Tables 5.6 and 5.7 show that a similar relationship holds for the PARP expected length criteria.

These three evaluations show that using the standard total route length (DARP) criteria can be misleading in terms of evaluating master routes. The DARP solution is about 10% better in terms of the standard total length criteria than the other two solutions. However, in terms of the multi-period (MARP) average route length criteria or the expected route length (PARP) criteria, the DARP solution is generally about two to five percent worse than the MARP or PARP solutions. These results confirm that using the standard deterministic-single period criteria of total route length is not a good way to evaluate the master routes since it does not take

| | DARP Solution (normalized length) | MARP Solution (normalized length) | PARP Solution (normalized length) |
|---|---|---|---|
| Service Route 1 | 1.00 | 0.9810 | 0.9795 |
| Service Route 2 | 1.00 | 0.9455 | 0.9577 |
| Service Route 3 | 1.00 | 0.9753 | 0.9799 |
| Service Route 4 | 1.00 | 0.9876 | 0.9925 |
| Service Route 5 | 1.00 | 0.9754 | 0.9892 |

Table 5.8: *Problem Sets Drawn From Actual Industrial Data: Master Route Quality using the MARP Objective Function.*

| | DARP Solution | MARP Solution | PARP Solution |
|---|---|---|---|
| Problem Set 1 | 11.0655 | 10.6726 | 10.7047 |
| Problem Set 2 | 11.579 | 11.1001 | 11.1584 |
| Problem Set 3 | 11.0916 | 10.5328 | 10.5557 |
| Problem Set 4 | 11.1758 | 10.7178 | 10.7421 |
| Problem Set 5 | 11.7098 | 11.2383 | 11.309 |

Table 5.9: *Computer Generated Problem Sets: Master Route Quality using the MARP Objective Function.*

into account the daily variation in customer demands.

The multi-period average route length or the probabilistic model's expected route length criteria are much better at accounting for the daily variation in customer demands. The results of Tables 5.6 to 5.9 indicate that both criteria are comparable in evaluating the quality of master routes. Since the MARP solution approach is much faster for problems of the sizes encountered in small package shipping firm practice (see Tables 5.2 and 5.3), the MARP solution approach may be preferred. For realistic sized problems, the PARP and its solution procedure may require excessive amounts of computation time.

## 5.7 Conclusions

We have considered the local routing problem for small package shipping where customer demands vary daily. In this context, node routing problems such as the probabilistic traveling salesman problem may not be appropriate since the set of customers served over a multi-day time horizon may be quite large. Instead, arc routing problems where a set of arcs instead of nodes must be traversed offer more tractable decision models. We introduced three types of arc routing problems and described heuristic solution approaches for two of them. Our computational results with test problems based on actual small package shipping firm data as well as on computer generated data confirm that the standard deterministic single period arc covering model does not produce the most desirable types of master routes. The multi-period and the probabilistic arc routing problems produce better master routes by taking into account daily customer demand variation via multi-time period or probabilistic models. Our computational results also show the multi-period model (with a moderate number of time periods) is much simpler to solve than the probabilistic model since it avoids the burden of expected length calculations required by the probabilistic model.

In the context of small package shipping firm planning operations, the deterministic arc routing problem (DARP) is convenient to use in obtaining a master route since it requires only a limited set of input parameters: the starting and ending locations, a set of arcs (street segments), the length of each street segment, the length of the shortest path between an endpoint of any segment to an endpoint of

any other segment, and the length of the shortest path from/to the starting/ending location to/from the endpoint of any street segment, and is a relatively simple model. However, our results indicate that noticeable (ranges from two to five percent) improvements can be obtained in master route quality by using a somewhat more complex model such as a multi-period or probabilistic arc routing problem that takes into account the daily variation in customer demand instead of the simpler deterministic and single period arc routing problem. As we have discussed previously, since small package local service operations occur on all weekdays throughout the entire United States, such improvements could represent substantial economic benefits in the cost of small package local service operations.

Our results suggest some interesting relationships among the arc routing models. If we wish to obtain a good solution for the PARP, we could produce the data for a MARP by using the presence probabilities to generate the sets of street segments that must be traversed during each time period. As we have seen, if the number of time periods is moderate, then the multi-day evaluation procedure is reasonably efficient in producing a master route. This approach to solving the PARP can also be utilized for other type of problems with an expected value objective function such as the probabilistic traveling salesman problem. We intend to pursue this possible new approach to solving the PTSP and related models as an area of future research. We intend to analyze this new approach and determine the number of time periods required for a multi-period model (e.g., MARP) to be a reasonable approximation to the probabilistic model (e.g., PARP) in terms of the master route produced. Note that this approach to finding solutions for probabilistic models can be viewed as an

optimization-simulation approach where each time period represents a simulation of the PARP and we use all of the simulations to obtain an optimization of the PARP.

There is also a reverse relationship between the PARP and MARP. We can take a given MARP and formulate a corresponding PARP by using the multi-period series of sets of street segments that must be traversed daily and use these data to compute presence probabilities for the PARP.

# Chapter 6

## Summary

In this dissertation we studied four network problems that arise in the transportation, telecommunications, and supply chain management application domains.

We investigated the split delivery vehicle routing problem (SDVRP) where a customer's demand can be split among several vehicles. We developed a new efficient heuristic that combined a mixed integer program and a record-to-record travel algorithm. This adds to the paradigm of combining metaheuristics and exact algorithms to solve combinatorial problems, which has recently received considerable attention from researchers as pointed out by Puchinger and Raidl (2005). In addition, computational results on the existing benchmark problems indicated that our algorithm outperformed the best-known solution approaches. Finally, we created 21 new test problems that can be used as benchmark problems. This helps researchers who are interested in studying the SDVRP as there are few test problems.

We studied the regenerator location problem (RLP) that arises in optical networks. We proved that the RLP is NP-complete and developed three heuristics that produced high-quality solutions for large-size networks in a matter of seconds. Specifically, on the 49 problem sets that we generated for the computational experiments, the MIP solver we used found the optimal solutions for eight problem sets within the time limit of 7200 seconds. For the remaining problem sets, the solver

provided the best lower bounds. In comparison, our heuristics found the optimal solutions for 21 problem sets (when the heuristic solution coincided with the lower bound we knew it was optimal). Additionally, we studied a generalization of the RLP and developed two efficient heuristics. We generated 42 problem sets for the computational experiment on the generalized RLP. The MIP solver found the optimal solutions for 29 problem sets while our heuristics found optimal solutions for 23 problem sets. The average computation time of the heuristics was over 100 times faster than that of the MIP solver.

We examined the parametric uncapacitated network design problems on series-parallel graphs, which has potential application in supply chain management. We described a polynomial time algorithm for the problem. Our results built upon the work of Wald (1999) and Raghavan et al. (2002) in the sense that we considered the bi-criteria network design problem on directed series-parallel graphs with single source and multiple sinks. To the best of our knowledge this is the first polynomial time algorithm for the problem.

We considered a problem that arises in the small package delivery. In practice, instead of re-optimizing the delivery routes on a daily basis (this can be quite time consuming), the service provider is encouraged to follow a master route — a pre-defined sequence of street segments— over a planning horizon. Currently, the well-known deterministic arc routing problem (DARP) model is often used to build the master routes. However, this approach ignores the uncertainty in the street segment presence probability — the probability that a street segment requires a visit on a particular day. We introduced two new models, namely the probabilistic arc

routing problem (PARP) model and the multi-period arc routing problem (MARP) model, which took into account the street segment presence probability. PARP attempted to minimize the expected length of the master route. It utilized a variation of the expected length formulation derived from Bertsimas et al. (1990). Consequently, it assumed that the street segment presence probabilities were independent. This model may require excessive amounts of computation time as the number of street segments can be quite large for a typical real-world problem. On the other hand, MARP tried to minimize average length of the master route over pre-specified time periods (days). This approximation significantly reduced the computational burden. Additionally, by utilizing the historical data over the pre-specified time periods, MARP incorporated real-world correlations among the street segment presence probabilities. Our computational results showed that PARP and MARP may produce more efficient master routes than DARP by taking into account the uncertainty in street segment presence.

# Appendix A

## Use Clarke-and-Wright Solutions as the EMIP Starting Solutions

The classical Clarke and Wright (1964) savings algorithm (CW) is perhaps the most widely known heuristic for the vehicle routing problem (VRP). The basic idea is to gradually build a feasible solution by combining two small routes into one larger route while attempting to keep the solution cost as low as possible. The merging operations between routes are ranked in terms of savings — a measure of the reduction in the travelling costs.

Specifically, consider a depot 0 and $n$ customers. Suppose that initially the solution to the VRP consists of using $n$ vehicles and assigning one vehicle to each one of the $n$ customers. The total travelling cost of this solution is $\sum 2l_{0,i}$. If now we merge two routes which currently serve i and j respectively, into one single route, the total travelling cost is reduced by the amount:

$$S_{i,j} = l_{0,i} + l_{0,j} - l_{i,j} \tag{A.1}$$

CW ranks all the feasible merging operations (a merging operation is feasible if the resulting route satisfies all the constraints of the VRP) in descending order according to $S_{i,j}$.

In the modified Clarke-and-Wright algorithm (Gaskell, 1967 and Yellow, 1970), a parameter $\lambda$ is introduced to control the relative importance of the direct arc between the two customers in the savings computation:

$$S_{i,j} = l_{0,i} + l_{0,j} - \lambda l_{i,j} \tag{A.2}$$

We use CW to generate the starting solution for EMIP because it is very fast and relatively easy to implement. Furthermore, we choose the classical CW instead of the modified CW for three reasons: (1) the former does not require parameter ($\lambda$) tuning; (2) as mentioned in Sec. 2.4, when customer demands are small most of the savings attribute to VRTR which generates good results regardless of the value for $\lambda$ (Li, Golden, and Wasil, 2005); (3) when customer demands are large the value for $\lambda$ has little effect on the starting solution. For instance, in scenario 6, $n$ vehicles are needed (one for each of the customers) no matter what value $\lambda$ is in the modified CW.

# Appendix B

## The Problem Generator for the New SDVRP Benchmark Problems

The problem generator and specifications for the 21 problems are given in the Figure B.1. Table B.1 lists the 21 problems and the corresponding parameter setting. Figure B.3 and B.4 list the data files for SD1 and SD2.

# Appendix C

## Data Files for Problem Set 1

Figure C.1 provides the street segment information for Problem Set 1. Table C.1 list the daily realized street segments over the 30-day study interval.

(x(i), y(i)) are the coordinates of customer i, where i = 0 is the depot. q(i) is the demand of customer i. A and B are parameters that determine the number of customers n, where n = A× B. A can be seen as the number of rays and B the number of layers. Vehicle capacity is 100 units. Customer demand is either 60 or 90 units. All data recorded to four decimal places.

```
begin
  ω = 0
  x(ω ) = 0, y( ω) = 0, q(ω ) = 0
  for k: = 1 to B do
    begin
      γ= 10k
      for i: = 1 to A do
        begin
          ω=ω+ 1
          x(ω) =γ cos[2(i − 1)π/A]
          y(ω) =γ sin[2(i − 1)π/A]
          if mod(i, 2) = 1
            then q(ω) = 60
            else q(ω) = 90
        end
    end
end.
```

Figure B.1: *Generator for New SDVRPs.*

| Problem | A | B | n |
|---|---|---|---|
| SD1 | 4 | 2 | 8 |
| SD2 | 4 | 4 | 16 |
| SD3 | 8 | 2 | 16 |
| SD4 | 12 | 2 | 24 |
| SD5 | 8 | 4 | 32 |
| SD6 | 16 | 2 | 32 |
| SD7 | 4 | 10 | 40 |
| SD8 | 4 | 12 | 48 |
| SD9 | 12 | 4 | 48 |
| SD10 | 16 | 4 | 64 |
| SD11 | 4 | 20 | 80 |
| SD12 | 8 | 10 | 80 |
| SD13 | 8 | 12 | 96 |
| SD14 | 12 | 10 | 120 |
| SD15 | 12 | 12 | 144 |
| SD16 | 72 | 2 | 144 |
| SD17 | 8 | 20 | 160 |
| SD18 | 16 | 10 | 160 |
| SD19 | 16 | 12 | 192 |
| SD20 | 12 | 20 | 240 |
| SD21 | 72 | 4 | 288 |

Table B.1: *The 21 New Test Problems.*

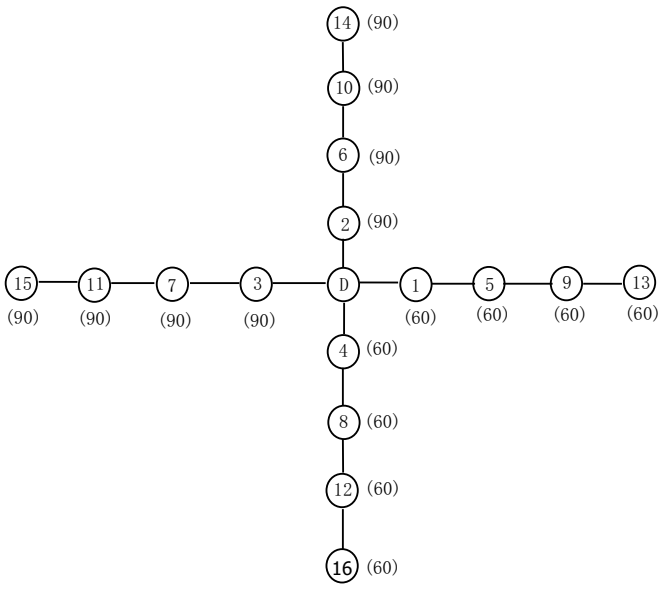



Figure B.2: *SD2 with 16 customers. The numbers besides the nodes are customer demands. The problem has four rays and four layers.*

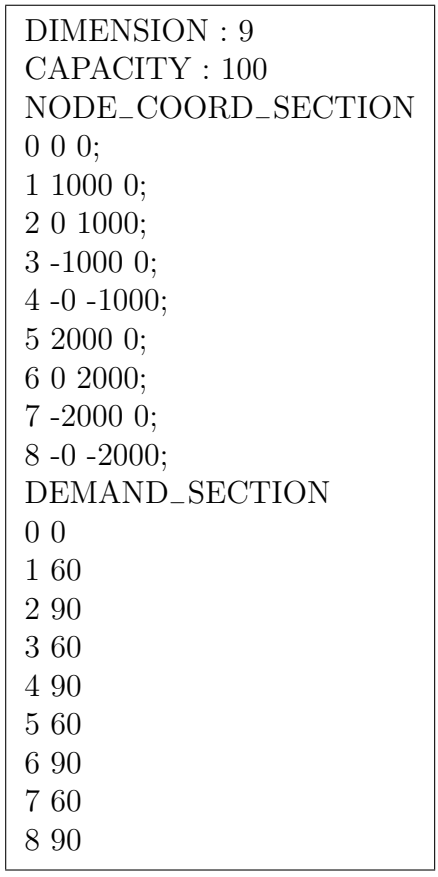



Figure B.3: *Data File for SD1.*

```
DIMENSION :17
CAPACITY :100
NODE_COORD_SECTION
0 0 0;
1 1000 0;
2 0 1000;
3 -1000 0;
4 0 -1000;
5 2000 0;
6 0 2000;
7 -2000 0;
8 0 -2000;
9 3000 0;
10 0 3000;
11 -3000 0;
12 0 -3000;
13 4000 0;
14 0 4000;
15 -4000 0;
16 -1 -4000;
DEMAND_SECTION
0 0
1 60
2 90
3 60
4 90
5 60
6 90
7 60
8 90
9 60
10 90
11 60
12 90
13 60
14 90
15 60
16 90
```

Figure B.4: *Data File for SD2.*

NUMBER OF STREET SEGMENTS: 200
1 24.5 0 24.5 0 1
2 10.6796 9.8892 11.8954 9.0479 0.597388
3 19.4382 16.7206 16.9158 19.8475 0.625739
4 30.4251 45.9839 32.0816 46.582 0.0660156
5 6.3956 23.5023 3.0884 22.6761 0.660773
6 7.5545 22.723 6.3956 23.5023 0.895575
7 9.4818 6.6707 12.3962 6.9702 0.865149
8 11.1892 7.6425 9.4818 6.6707 0.484869
9 3.1555 38.9832 6.3762 40.7707 0.438208
10 0.6866 23.3551 1.6327 27.2386 0.209906
11 1.6327 27.2386 3.5409 27.5424 0.440985
12 3.5409 27.5424 4.8239 26.1375 0.687537
13 39.0213 6.6132 38.743 5.2584 0.0817322
14 40.274 18.4128 40.1624 20.554 0.763434
15 40.1624 20.554 36.1552 18.4801 0.722662
16 6.3263 48.2132 8.7318 43.9014 0.421393
17 33.5709 28.4897 29.2036 29.3898 0.5315
18 29.2036 29.3898 33.8051 29.0527 0.472693
19 33.8051 29.0527 29.309 30.6971 0.42417
20 9.6901 10.3913 9.8862 9.3872 0.450934
21 34.665 33.6404 37.0453 29.3213 0.542853
22 25.1034 32.0194 26.7761 31.0165 0.330328
23 22.6721 28.677 25.1034 32.0194 0.69068
24 23.7811 28.6773 22.6721 28.677 0.414252
25 23.3932 38.7833 22.7404 40.9783 0.363196
26 23.3551 36.8393 22.4168 38.8425 0.719763
27 22.4168 38.8425 23.6514 37.8699 0.580756
28 23.6514 37.8699 24.9111 36.8144 0.149542
29 35.611 11.0092 34.738 9.0118 0.23075
30 20.694 38.0676 18.2671 34.9871 0.638708
31 18.2671 34.9871 18.5812 33.2783 0.831519
32 41.7464 31.3044 38.6047 34.4482 0.0645203
33 31.0654 29.8309 29.0535 27.7445 0.332251
34 29.0535 27.7445 28.374 26.625 0.35993
35 28.374 26.625 31.7282 26.4959 0.0991272
36 31.7841 3.5156 34.2957 1.6107 0.696875
37 19.4471 32.9996 20.2103 35.4568 0.0552063
38 16.7686 32.8863 19.4471 32.9996 0.649176
39 16.4324 30.9704 16.7686 32.8863 0.832709
40 13.8687 24.028 15.8617 22.4307 0.265814
41 15.8617 22.4307 12.4979 22.9504 0.888647
Cont. on next page.

42 21.1566 36.1683 16.3376 35.7834 0.0194702
43 33.3532 31.2569 34.7961 29.1504 0.670935
44 35.0752 27.6746 33.3532 31.2569 0.634161
45 49.7009 8.6014 48.3272 6.9937 0.496252
46 4.2893 31.6193 6.1234 29.1779 0.00656738
47 30.2881 31.4338 31.1691 34.3948 0.0299744
48 29.1124 35.1549 30.2881 31.4338 0.0598206
49 8.9273 8.9942 7.4097 10.4111 0.0549072
50 12.3264 7.7648 8.9273 8.9942 0.492224
51 14.1217 5.2003 12.3264 7.7648 0.22785
52 39.4298 29.456 43.1259 27.2659 0.175482
53 29.5059 16.0965 27.3878 14.1679 0.328253
54 24.8877 39.1085 22.7646 43.4479 0.515509
55 23.367 40.783 24.8877 39.1085 0.554053
56 1.358 21.7438 0.0076 20.5433 0.452826
57 10.1523 27.4571 7.8629 28.4546 0.493506
58 9.4473 26.3983 10.1523 27.4571 0.135535
59 29.9423 6.8008 26.73 4.589 0.624976
60 26.73 4.589 24.4636 7.0114 0.896948
61 24.4636 7.0114 22.969 3.0109 0.814093
62 14.1349 11.3449 15.419 8.8608 0.0416321
63 14.4551 8.2597 14.1349 11.3449 0.422736
64 6.9113 18.5152 6.3965 19.989 0.22016
65 8.502 18.0172 6.9113 18.5152 0.0244751
66 9.4724 18.3213 8.502 18.0172 0.842444
67 19.0372 12.8329 20.433 15.1276 0.897253
68 50.7413 5.1256 46.9467 7.8125 0.39176
69 52.4467 5.9303 50.7413 5.1256 0.0799866
70 29.4678 45.0268 29.6387 43.8171 0.360846
71 37.4176 8.3176 37.1089 7.3655 0.561163
72 37.1089 7.3655 39.3536 6.2355 0.0541687
73 27.309 14.221 29.3991 13.6703 0.373328
74 26.7229 9.7897 27.309 14.221 0.245734
75 25.1614 11.7192 26.7229 9.7897 0.624915
76 38.787 14.6493 35.5896 14.7507 0.632391
77 38.6108 15.6951 38.787 14.6493 0.109778
78 15.7519 6.7593 18.7958 6.5476 0.237585
79 27.7167 30.3336 29.6936 33.1894 0.671454
80 27.7167 30.3336 29.833 31.5718 0.75025
81 29.833 31.5718 28.889 34.4215 0.514349
82 16.5741 22.4594 12.6422 23.2237 0.763617
83 12.6422 23.2237 13.5846 20.9799 0.0221558
Cont. on next page.

84 13.5846 20.9799 12.108 23.7918 0.428107
85 43.6325 14.856 42.7429 16.7557 0.31864
86 43.2655 13.3647 43.6325 14.856 0.181311
87 22.0465 4.4955 22.5662 1.8158 0.0211548
88 7.9666 34.6436 7.5834 35.6875 0.123602
89 44.7952 21.5319 45.4819 17.6102 0.213293
90 43.6618 23.0528 44.7952 21.5319 0.887823
91 41.7984 26.5739 43.6618 23.0528 0.656348
92 0.3356 19.7892 0.2716 22.6379 0.1724
93 0.9808 15.8648 0.3356 19.7892 0.721899
94 24.4307 15.0534 24.115 16.1865 0.101599
95 32.8192 44.6184 37.2849 43.3731 0.113074
96 10.4691 40.5548 7.0208 43.5157 0.00274658
97 7.0208 43.5157 9.2827 40.0583 0.0570435
98 9.2827 40.0583 6.4923 38.162 0.624426
99 23.7762 18.6646 24.1476 15.6013 0.617621
100 24.1476 15.6013 20.7838 18.9546 0.14054
101 17.6634 7.4772 20.6406 8.7219 0.00751343
102 18.4783 6.3602 17.6634 7.4772 0.282904
103 27.563 47.4687 30.6259 46.8994 0.367346
104 15.2847 40.741 16.6709 41.1879 0.0415039
105 19.5235 39.1749 21.0617 38.4308 0.62818
106 19.5235 39.1749 21.299 36.5523 0.219031
107 0.5253 28.1046 1.7563 31.9412 0.86463
108 19.2032 0.0671 16.9384 0.4896 0.0143127
109 51.1825 34.9606 48.8281 35.6003 0.760443
110 42.0547 2.6215 37.4805 4.2503 0.215643
111 37.5824 19.693 42.3557 20.6357 0.117041
112 21.4989 11.3337 20.3552 12.0056 0.816046
113 16.686 11.4787 21.4989 11.3337 0.57804
114 33.6791 2.3741 32.3212 1.1581 0.0252747
115 32.4697 34.6258 30.8548 36.1725 0.757941
116 33.658 30.1316 32.4697 34.6258 0.769598
117 31.0896 33.3902 33.658 30.1316 0.796149
118 3.5797 0.9262 4.493 0.3895 0.340796
119 9.8358 36.6867 12.1325 36.7154 0.584357
120 43.222 17.5461 44.4703 15.8135 0.50022
121 44.4703 15.8135 40.6547 17.532 0.196173
122 40.6547 17.532 41.4943 15.4641 0.349158
123 6.7673 27.6459 2.5228 27.5218 0.788062
124 47.2362 3.2036 43.3945 2.7573 0.252753
125 41.861 14.6149 39.348 15.0268 0.090094
Cont. on next page.

126 39.348 15.0268 41.5467 12.543 0.423895
127 41.5467 12.543 43.0301 12.3616 0.107306
128 25.6409 39.7974 23.2224 39.4135 0.165778
129 26.1738 38.9467 25.6409 39.7974 0.0177979
130 35.5286 12.4649 33.4925 12.4455 0.00305176
131 8.215 24.2065 10.3928 20.4422 0.699438
132 33.0358 26.4711 33.1161 28.4271 0.869543
133 33.0246 22.0318 33.0358 26.4711 0.119727
134 30.1233 20.3458 33.0246 22.0318 0.750952
135 45.1904 38.7253 44.7291 34.1679 0.798468
136 4.4586 37.7747 2.6494 38.5388 0.0305176
137 37.1434 3.3879 32.6508 2.5757 0.234473
138 30.304 11.9843 30.8051 10.8075 0.518591
139 30.8051 10.8075 34.8231 11.0705 0.291937
140 34.8231 11.0705 36.4531 7.9654 0.633856
141 5.4618 29.0206 5.9692 25.4059 0.00232544
142 7.4538 29.7222 5.4618 29.0206 0.526678
143 48.262 36.8555 49.4354 37.3642 0.517828
144 44.9126 34.7103 48.262 36.8555 0.180396
145 42.74 34.2473 44.9126 34.7103 0.677924
146 7.4936 32.8293 7.3544 34.8467 0.163702
147 45.2469 31.192 42.5927 30.3004 0.111182
148 42.5927 30.3004 45.7405 27.8394 0.447699
149 1.8314 38.9175 3.5095 34.9823 0.688727
150 2.2253 37.9885 1.8314 38.9175 0.221472
151 15.0055 36.7645 17.9022 38.2959 0.138373
152 17.9022 38.2959 19.1104 38.0528 0.119025
153 9.1797 39.1776 10.7376 38.4949 0.0597595
154 8.6389 40.4089 9.1797 39.1776 0.048584
155 4.6189 4.7795 3.7918 1.149 0.332953
156 4.0841 3.928 4.6189 4.7795 0.234961
157 28.5797 21.4496 30.8563 23.2727 0.0189514
158 30.4194 22.4896 28.5797 21.4496 0.881323
159 27.0641 25.1365 30.4194 22.4896 0.356909
160 34.4757 46.3837 32.8391 46.2381 0.41593
161 32.8391 46.2381 36.3884 46.7173 0.710181
162 42.0166 21.7163 41.6449 16.9789 0.778143
163 44.0283 17.9959 41.6449 16.9789 0.827399
164 32.9666 2.4384 32.0016 3.7811 0.782263
165 32.0016 3.7811 33.2352 4.4693 0.0246338
166 33.2352 4.4693 30.7645 6.2165 0.412909
167 44.6272 4.0029 44.2825 6.0364 0.336951
Cont. on next page.

168 33.6456 48.6221 34.4125 47.2313 0.0877075
169 23.3215 37.3901 21.5609 34.2913 0.0916748
170 21.5609 34.2913 25.1738 31.5393 0.338446
171 25.1738 31.5393 21.3846 29.1262 0.19538
172 20.8572 15.4678 18.2298 15.1373 0.655463
173 15.5914 31.9962 14.6944 31.406 0.794012
174 40.0238 42.4423 39.6858 40.4112 0.763892
175 39.6858 40.4112 43.5909 39.7996 0.630255
176 29.4042 34.7838 25.8087 36.496 0.887091
177 29.4042 34.7838 30.1087 31.6635 0.206885
178 30.1087 31.6635 30.0516 28.4061 0.429694
179 39.4045 11.0949 37.1277 14.5386 0.0144348
180 36.2076 14.3608 39.4045 11.0949 0.785803
181 22.0291 6.5369 19.194 10.5592 0.254584
182 22.847 8.0551 22.7965 6.4224 0.731268
183 17.8314 20.0562 20.7624 21.7363 0.699072
184 20.7624 21.7363 24.3084 20.3288 0.208136
185 24.3084 20.3288 22.8634 20.4819 0.312537
186 45.3857 22.0627 48.1625 20.7644 0.456915
187 46.6125 10.9009 49.5972 9.0602 0.070105
188 46.1396 7.9423 47.5494 8.4518 0.188055
189 46.1396 7.9423 47.0882 5.9241 0.289984
190 47.0882 5.9241 49.0822 2.4165 0.702551
191 18.7027 36.7188 18.0142 38.0522 0.678595
192 18.0142 38.0522 18.4155 36.3494 0.0270752
193 18.4155 36.3494 21.7113 36.1122 0.138922
194 7.45 28.4744 9.6741 30.5695 0.361395
195 6.8977 26.8167 7.45 28.4744 0.413397
196 3.2723 35.7207 2.002 39.3112 0.0741943
197 4.9427 34.8144 3.2723 35.7207 0.881659
198 1.2308 32.5429 2.7869 32.6377 0.0895142
199 2.9836 36.4381 5.8448 32.3635 0.898138
200 25.5 0 25.5 0 1

Figure C.1: *Problem Set 1 has 200 street segments. The first column is the index for the street segment. The X/Y coordinates for the two endpoints are listed in the (second, forth)/(third, fifth) columns. The sixth column lists the street segment presence probability.*

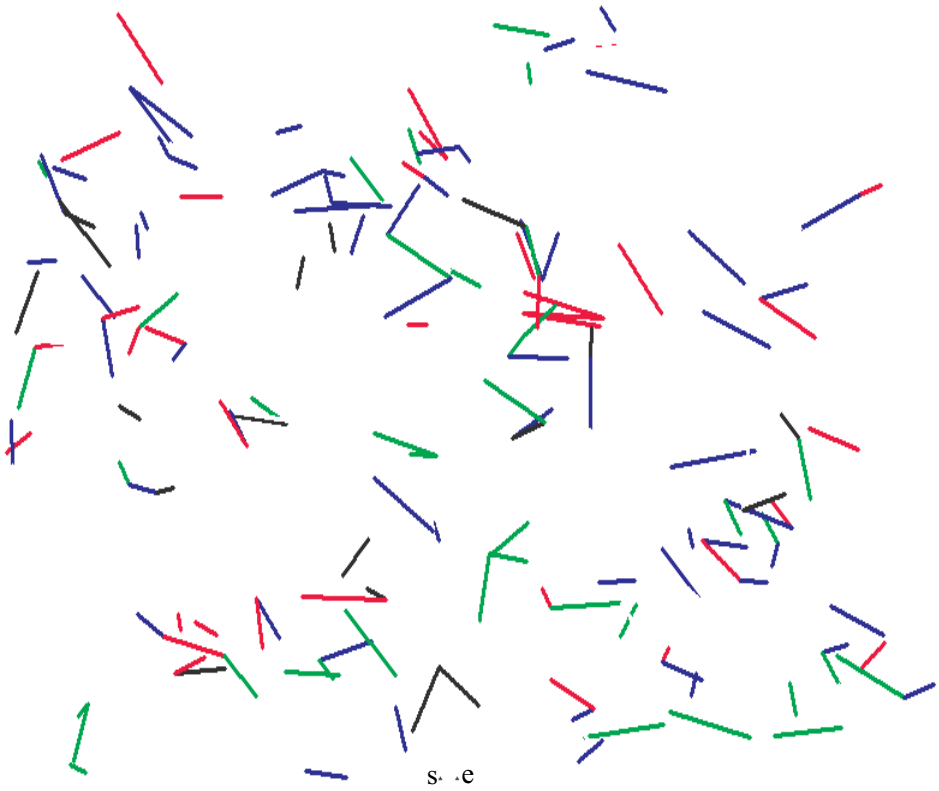



Figure C.2: *Illustration for Problem Set 1. s is the starting location and e is the ending location.*

| Day1 | Day2 | Day3 | Day4 | Day5 | Day6 | Day7 | Day8 | Day9 | Day10 |
|---|---|---|---|---|---|---|---|---|---|
| 50 | 51 | 20 | 51 | 2 | 50 | 20 | 20 | 50 | 51 |
| 156 | 50 | 2 | 50 | 7 | 8 | 8 | 8 | 20 | 50 |
| 7 | 20 | 8 | 20 | 63 | 7 | 155 | 7 | 118 | 20 |
| 63 | 2 | 7 | 7 | 113 | 182 | 156 | 63 | 156 | 2 |
| 78 | 8 | 63 | 78 | 112 | 113 | 63 | 62 | 63 | 8 |
| 113 | 118 | 112 | 182 | 67 | 67 | 102 | 102 | 78 | 155 |
| 67 | 7 | 67 | 113 | 172 | 172 | 113 | 182 | 182 | 7 |
| 183 | 182 | 183 | 67 | 3 | 3 | 112 | 181 | 113 | 63 |
| 99 | 113 | 185 | 172 | 184 | 183 | 67 | 113 | 112 | 78 |
| 41 | 112 | 82 | 3 | 99 | 99 | 82 | 184 | 172 | 113 |
| 57 | 67 | 41 | 183 | 94 | 82 | 41 | 99 | 3 | 112 |
| 107 | 172 | 57 | 99 | 82 | 41 | 84 | 82 | 183 | 67 |
| 197 | 185 | 123 | 94 | 41 | 84 | 197 | 41 | 82 | 172 |
| 98 | 82 | 107 | 82 | 84 | 107 | 98 | 84 | 41 | 3 |
| 199 | 41 | 197 | 84 | 123 | 197 | 149 | 123 | 123 | 183 |
| 142 | 84 | 9 | 57 | 107 | 9 | 199 | 107 | 107 | 99 |
| 131 | 123 | 149 | 123 | 197 | 149 | 66 | 197 | 197 | 41 |
| 66 | 9 | 142 | 107 | 98 | 131 | 6 | 98 | 98 | 84 |
| 5 | 16 | 131 | 197 | 149 | 66 | 93 | 149 | 150 | 123 |
| 56 | 98 | 6 | 98 | 199 | 6 | 195 | 199 | 149 | 107 |
| 93 | 199 | 56 | 150 | 142 | 5 | 119 | 142 | 199 | 197 |
| 10 | 131 | 93 | 149 | 66 | 56 | 31 | 131 | 142 | 98 |
| 12 | 66 | 10 | 199 | 6 | 11 | 173 | 66 | 131 | 199 |
| 195 | 6 | 11 | 66 | 5 | 12 | 39 | 6 | 66 | 66 |
| 119 | 56 | 12 | 64 | 11 | 146 | 38 | 92 | 64 | 6 |
| 30 | 93 | 194 | 5 | 12 | 105 | 191 | 195 | 6 | 5 |
| 31 | 11 | 106 | 56 | 195 | 30 | 27 | 194 | 5 | 93 |
| 173 | 12 | 105 | 93 | 105 | 31 | 28 | 119 | 56 | 12 |
| 39 | 194 | 30 | 194 | 30 | 173 | 23 | 105 | 93 | 88 |
| 38 | 105 | 173 | 152 | 31 | 39 | 80 | 39 | 11 | 119 |
| 191 | 30 | 39 | 105 | 173 | 191 | 117 | 38 | 12 | 106 |
| 26 | 31 | 191 | 30 | 38 | 26 | 116 | 26 | 194 | 105 |
| 28 | 39 | 26 | 31 | 27 | 27 | 115 | 23 | 30 | 30 |
| 23 | 38 | 169 | 173 | 23 | 23 | 178 | 80 | 31 | 31 |
| 116 | 191 | 170 | 39 | 22 | 22 | 33 | 81 | 173 | 39 |
| 115 | 26 | 24 | 191 | 80 | 117 | 134 | 48 | 39 | 38 |
| 177 | 27 | 23 | 27 | 81 | 116 | 132 | 116 | 38 | 37 |
| 33 | 28 | 80 | 170 | 117 | 115 | 44 | 178 | 191 | 191 |
| 34 | 24 | 116 | 24 | 177 | 177 | 43 | 158 | 193 | 26 |
| 159 | 23 | 115 | 23 | 178 | 33 | 79 | 134 | 26 | 27 |
| 158 | 80 | 33 | 81 | 33 | 158 | 176 | 132 | 24 | 169 |
| 134 | 81 | 34 | 117 | 35 | 18 | 55 | 17 | 80 | 170 |
| 44 | 117 | 159 | 116 | 158 | 44 | 54 | 18 | 81 | 24 |

Cont. on next page.

| Day1 | Day2 | Day3 | Day4 | Day5 | Day6 | Day7 | Day8 | Day9 | Day10 |
|---|---|---|---|---|---|---|---|---|---|
| 43 | 115 | 158 | 115 | 132 | 43 | 161 | 44 | 117 | 80 |
| 19 | 178 | 134 | 158 | 17 | 19 | 160 | 43 | 116 | 81 |
| 79 | 33 | 132 | 134 | 79 | 79 | 174 | 176 | 115 | 117 |
| 176 | 34 | 43 | 132 | 176 | 176 | 135 | 25 | 134 | 116 |
| 25 | 158 | 19 | 44 | 128 | 54 | 145 | 55 | 132 | 158 |
| 55 | 132 | 79 | 43 | 54 | 161 | 109 | 54 | 17 | 134 |
| 103 | 17 | 176 | 19 | 103 | 168 | 148 | 103 | 18 | 132 |
| 174 | 44 | 25 | 79 | 161 | 175 | 52 | 160 | 43 | 18 |
| 175 | 19 | 174 | 176 | 160 | 143 | 91 | 174 | 79 | 44 |
| 135 | 79 | 135 | 54 | 95 | 109 | 90 | 175 | 70 | 43 |
| 145 | 176 | 145 | 161 | 174 | 91 | 89 | 135 | 161 | 79 |
| 109 | 55 | 109 | 174 | 175 | 90 | 163 | 145 | 95 | 176 |
| 21 | 161 | 148 | 175 | 109 | 163 | 126 | 143 | 174 | 128 |
| 90 | 160 | 91 | 135 | 148 | 126 | 85 | 109 | 175 | 25 |
| 125 | 174 | 90 | 143 | 21 | 86 | 120 | 147 | 135 | 55 |
| 126 | 135 | 186 | 109 | 90 | 120 | 14 | 148 | 145 | 161 |
| 127 | 144 | 89 | 21 | 163 | 14 | 162 | 91 | 143 | 174 |
| 14 | 52 | 15 | 52 | 122 | 15 | 76 | 90 | 90 | 175 |
| 15 | 91 | 162 | 91 | 126 | 162 | 75 | 186 | 126 | 145 |
| 162 | 90 | 180 | 90 | 14 | 76 | 138 | 163 | 120 | 143 |
| 77 | 89 | 73 | 163 | 15 | 180 | 140 | 122 | 14 | 21 |
| 180 | 163 | 29 | 122 | 76 | 75 | 71 | 120 | 162 | 52 |
| 53 | 121 | 167 | 120 | 180 | 73 | 68 | 111 | 180 | 90 |
| 138 | 14 | 45 | 121 | 75 | 139 | 189 | 162 | 73 | 163 |
| 140 | 15 | 189 | 14 | 74 | 140 | 190 | 76 | 138 | 85 |
| 167 | 162 | 190 | 15 | 29 | 68 | 36 | 180 | 139 | 120 |
| 190 | 76 | 124 | 162 | 140 | 190 | 164 | 74 | 29 | 121 |
| 36 | 180 | 36 | 53 | 71 | 164 | 59 | 29 | 140 | 14 |
| 164 | 75 | 164 | 75 | 167 | 60 | 60 | 140 | 71 | 15 |
| 59 | 73 | 60 | 73 | 190 |  |  | 71 | 188 | 76 |
| 60 | 138 | 61 | 138 | 137 |  |  | 13 | 45 | 180 |
|  | 45 |  | 140 | 164 |  |  | 167 | 69 | 75 |
|  | 68 |  | 71 | 59 |  |  | 45 | 190 | 138 |
|  | 190 |  | 167 | 60 |  |  | 189 | 124 | 71 |
|  | 36 |  | 68 | 61 |  |  | 137 | 36 | 188 |
|  | 164 |  | 190 |  |  |  | 36 | 164 | 45 |
|  | 166 |  | 124 |  |  |  | 164 | 60 | 110 |
|  | 59 |  | 110 |  |  |  | 166 | 61 | 36 |
|  | 60 |  | 137 |  |  |  | 59 |  | 164 |
|  | 61 |  | 36 |  |  |  | 60 |  | 59 |
|  |  |  | 59 |  |  |  | 61 |  | 61 |
|  |  |  | 60 |  |  |  |  |  |  |
|  |  |  | 61 |  |  |  |  |  |  |

Cont. on next page.

| Day11 | Day12 | Day13 | Day14 | Day15 | Day16 | Day17 | Day18 | Day19 | Day20 |
|---|---|---|---|---|---|---|---|---|---|
| 50 | 2 | 2 | 50 | 20 | 20 | 51 | 50 | 155 | 2 |
| 8 | 118 | 8 | 20 | 2 | 8 | 50 | 7 | 7 | 8 |
| 182 | 156 | 118 | 2 | 102 | 118 | 118 | 63 | 182 | 118 |
| 112 | 7 | 156 | 8 | 182 | 7 | 7 | 182 | 67 | 156 |
| 67 | 102 | 7 | 118 | 112 | 63 | 63 | 181 | 3 | 7 |
| 172 | 182 | 182 | 7 | 67 | 102 | 182 | 112 | 183 | 112 |
| 183 | 112 | 181 | 102 | 184 | 182 | 112 | 67 | 185 | 67 |
| 184 | 67 | 112 | 182 | 99 | 113 | 67 | 82 | 99 | 172 |
| 99 | 172 | 67 | 113 | 100 | 112 | 183 | 40 | 100 | 3 |
| 82 | 99 | 3 | 112 | 82 | 67 | 99 | 41 | 82 | 183 |
| 40 | 41 | 183 | 67 | 41 | 172 | 100 | 123 | 40 | 41 |
| 41 | 57 | 184 | 3 | 57 | 3 | 82 | 107 | 41 | 57 |
| 123 | 123 | 82 | 185 | 123 | 185 | 40 | 197 | 107 | 123 |
| 107 | 107 | 41 | 82 | 107 | 99 | 41 | 149 | 196 | 107 |
| 197 | 197 | 84 | 40 | 197 | 94 | 57 | 199 | 149 | 198 |
| 16 | 9 | 123 | 41 | 153 | 40 | 123 | 142 | 199 | 197 |
| 98 | 16 | 107 | 84 | 149 | 123 | 198 | 66 | 131 | 196 |
| 150 | 149 | 197 | 123 | 199 | 107 | 197 | 6 | 66 | 9 |
| 199 | 199 | 9 | 107 | 142 | 9 | 9 | 5 | 6 | 154 |
| 142 | 142 | 97 | 16 | 131 | 98 | 150 | 93 | 5 | 16 |
| 131 | 131 | 98 | 98 | 66 | 149 | 149 | 11 | 93 | 199 |
| 66 | 66 | 149 | 199 | 6 | 199 | 199 | 12 | 92 | 131 |
| 64 | 6 | 199 | 142 | 56 | 131 | 131 | 194 | 12 | 66 |
| 6 | 5 | 131 | 131 | 93 | 66 | 6 | 106 | 194 | 64 |
| 5 | 56 | 6 | 66 | 11 | 6 | 5 | 30 | 105 | 6 |
| 56 | 93 | 56 | 6 | 119 | 5 | 12 | 31 | 30 | 5 |
| 92 | 12 | 93 | 93 | 105 | 93 | 119 | 173 | 31 | 93 |
| 11 | 195 | 195 | 11 | 30 | 10 | 31 | 39 | 39 | 10 |
| 12 | 146 | 194 | 12 | 173 | 31 | 173 | 38 | 38 | 11 |
| 195 | 105 | 119 | 88 | 39 | 173 | 191 | 191 | 24 | 12 |
| 119 | 31 | 151 | 119 | 191 | 39 | 26 | 26 | 22 | 195 |
| 106 | 173 | 105 | 105 | 22 | 38 | 170 | 27 | 80 | 119 |
| 105 | 39 | 30 | 30 | 80 | 26 | 171 | 28 | 81 | 151 |
| 31 | 191 | 31 | 31 | 117 | 27 | 23 | 23 | 117 | 105 |
| 173 | 26 | 173 | 173 | 116 | 24 | 80 | 22 | 116 | 30 |
| 39 | 27 | 39 | 39 | 159 | 23 | 81 | 80 | 115 | 31 |
| 38 | 170 | 26 | 38 | 158 | 80 | 117 | 115 | 33 | 191 |
| 26 | 80 | 171 | 191 | 134 | 81 | 115 | 33 | 34 | 193 |
| 27 | 81 | 24 | 26 | 132 | 117 | 159 | 34 | 158 | 26 |
| 24 | 117 | 23 | 171 | 17 | 116 | 158 | 159 | 134 | 170 |
| 22 | 116 | 22 | 23 | 44 | 115 | 134 | 158 | 132 | 23 |
| 80 | 115 | 80 | 80 | 43 | 177 | 133 | 134 | 44 | 22 |

Cont. on next page.

| Day11 | Day12 | Day13 | Day14 | Day15 | Day16 | Day17 | Day18 | Day19 | Day20 |
|---|---|---|---|---|---|---|---|---|---|
| 117 | 159 | 116 | 117 | 176 | 158 | 132 | 132 | 43 | 80 |
| 116 | 158 | 115 | 116 | 161 | 133 | 17 | 44 | 19 | 116 |
| 115 | 134 | 178 | 178 | 160 | 18 | 44 | 43 | 176 | 115 |
| 159 | 132 | 34 | 159 | 95 | 43 | 43 | 19 | 25 | 177 |
| 158 | 17 | 159 | 132 | 175 | 79 | 79 | 176 | 55 | 178 |
| 134 | 79 | 158 | 18 | 145 | 176 | 176 | 128 | 103 | 134 |
| 132 | 176 | 132 | 43 | 143 | 55 | 55 | 25 | 161 | 132 |
| 18 | 55 | 18 | 79 | 109 | 54 | 54 | 174 | 160 | 17 |
| 43 | 54 | 44 | 176 | 21 | 70 | 161 | 135 | 174 | 19 |
| 55 | 70 | 43 | 25 | 91 | 161 | 135 | 145 | 175 | 79 |
| 103 | 161 | 79 | 55 | 90 | 175 | 143 | 143 | 135 | 176 |
| 161 | 145 | 176 | 161 | 186 | 135 | 109 | 21 | 109 | 25 |
| 168 | 143 | 55 | 174 | 89 | 145 | 21 | 91 | 148 | 55 |
| 160 | 109 | 70 | 175 | 163 | 143 | 91 | 90 | 21 | 54 |
| 135 | 148 | 161 | 143 | 14 | 109 | 90 | 163 | 91 | 70 |
| 145 | 91 | 160 | 109 | 15 | 147 | 186 | 126 | 163 | 161 |
| 109 | 163 | 174 | 21 | 76 | 148 | 89 | 86 | 126 | 174 |
| 148 | 127 | 135 | 91 | 138 | 91 | 163 | 120 | 120 | 135 |
| 90 | 14 | 109 | 90 | 140 | 90 | 126 | 14 | 14 | 145 |
| 186 | 76 | 147 | 163 | 13 | 163 | 85 | 15 | 15 | 143 |
| 122 | 180 | 148 | 120 | 45 | 122 | 121 | 162 | 162 | 109 |
| 126 | 53 | 90 | 14 | 190 | 125 | 14 | 180 | 76 | 21 |
| 120 | 75 | 186 | 15 | 137 | 127 | 111 | 75 | 53 | 90 |
| 121 | 74 | 163 | 162 | 36 | 86 | 162 | 138 | 75 | 186 |
| 15 | 73 | 85 | 76 | 164 | 85 | 76 | 140 | 73 | 15 |
| 162 | 138 | 120 | 180 | 166 | 15 | 180 | 68 | 138 | 180 |
| 77 | 139 | 14 | 74 | 60 | 180 | 53 | 36 | 140 | 75 |
| 180 | 167 | 15 | 140 | 61 | 53 | 75 | 164 | 71 | 139 |
| 53 | 189 | 162 | 71 |  | 140 | 73 | 59 | 190 | 140 |
| 75 | 36 | 76 | 45 |  | 167 | 140 | 60 | 36 | 188 |
| 139 | 166 | 180 | 190 |  | 36 | 167 | 61 | 164 | 189 |
| 140 | 60 | 53 | 164 |  | 164 | 45 |  | 60 | 36 |
| 188 | 61 | 73 | 166 |  | 60 | 190 |  | 61 | 166 |
| 69 |  | 71 | 60 |  | 61 | 137 |  |  | 59 |
| 68 |  | 189 | 61 |  |  | 164 |  |  | 60 |
| 190 |  | 124 |  |  |  | 166 |  |  |  |
| 164 |  | 36 |  |  |  | 59 |  |  |  |
| 166 |  | 164 |  |  |  | 60 |  |  |  |
| 59 |  | 59 |  |  |  | 61 |  |  |  |
| 60 |  | 60 |  |  |  |  |  |  |  |
| 61 |  | 61 |  |  |  |  |  |  |  |

Cont. on next page.

| Day21 | Day22 | Day23 | Day24 | Day25 | Day26 | Day27 | Day28 | Day29 | Day30 |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 50 | 49 | 155 | 50 | 155 | 50 | 51 | 50 | 51 |
| 2 | 2 | 2 | 7 | 2 | 118 | 2 | 20 | 2 | 20 |
| 181 | 155 | 8 | 182 | 155 | 7 | 8 | 2 | 8 | 2 |
| 112 | 7 | 155 | 113 | 7 | 78 | 7 | 155 | 7 | 118 |
| 172 | 78 | 156 | 112 | 78 | 102 | 63 | 7 | 102 | 7 |
| 3 | 181 | 7 | 67 | 182 | 113 | 182 | 182 | 182 | 63 |
| 183 | 113 | 182 | 172 | 113 | 112 | 112 | 113 | 67 | 182 |
| 82 | 112 | 181 | 3 | 112 | 172 | 67 | 112 | 172 | 181 |
| 41 | 67 | 112 | 185 | 67 | 183 | 172 | 67 | 3 | 113 |
| 84 | 172 | 67 | 99 | 172 | 185 | 3 | 172 | 183 | 112 |
| 58 | 3 | 183 | 82 | 3 | 99 | 183 | 3 | 82 | 67 |
| 57 | 183 | 99 | 41 | 183 | 100 | 185 | 183 | 41 | 172 |
| 123 | 82 | 57 | 58 | 99 | 82 | 99 | 99 | 84 | 3 |
| 107 | 41 | 123 | 57 | 82 | 41 | 40 | 82 | 57 | 183 |
| 197 | 58 | 197 | 107 | 84 | 107 | 41 | 41 | 123 | 184 |
| 9 | 57 | 9 | 197 | 107 | 197 | 123 | 58 | 107 | 185 |
| 98 | 123 | 16 | 16 | 197 | 9 | 107 | 57 | 197 | 41 |
| 149 | 197 | 150 | 150 | 16 | 16 | 197 | 123 | 9 | 57 |
| 199 | 98 | 199 | 199 | 98 | 98 | 9 | 107 | 16 | 123 |
| 142 | 142 | 142 | 131 | 149 | 149 | 149 | 197 | 98 | 107 |
| 66 | 131 | 131 | 66 | 66 | 199 | 199 | 16 | 199 | 197 |
| 6 | 6 | 66 | 6 | 6 | 66 | 131 | 98 | 142 | 16 |
| 5 | 12 | 6 | 93 | 5 | 64 | 6 | 149 | 66 | 149 |
| 56 | 88 | 5 | 106 | 11 | 56 | 5 | 199 | 6 | 199 |
| 92 | 119 | 93 | 105 | 119 | 93 | 93 | 142 | 5 | 131 |
| 11 | 151 | 10 | 31 | 30 | 92 | 10 | 66 | 56 | 66 |
| 195 | 105 | 11 | 38 | 173 | 12 | 119 | 64 | 93 | 6 |
| 194 | 31 | 12 | 171 | 39 | 195 | 30 | 12 | 12 | 5 |
| 119 | 173 | 195 | 23 | 26 | 119 | 173 | 119 | 119 | 93 |
| 106 | 39 | 194 | 22 | 27 | 104 | 191 | 152 | 31 | 12 |
| 31 | 38 | 146 | 80 | 24 | 30 | 26 | 105 | 173 | 195 |
| 38 | 27 | 31 | 81 | 23 | 31 | 170 | 30 | 39 | 146 |
| 27 | 170 | 173 | 117 | 81 | 173 | 171 | 173 | 191 | 151 |
| 24 | 80 | 39 | 115 | 117 | 39 | 24 | 39 | 193 | 152 |
| 81 | 81 | 38 | 178 | 116 | 38 | 23 | 38 | 26 | 105 |
| 117 | 115 | 191 | 158 | 115 | 191 | 80 | 191 | 27 | 30 |
| 116 | 158 | 193 | 134 | 33 | 26 | 117 | 26 | 23 | 31 |
| 178 | 134 | 26 | 132 | 158 | 27 | 115 | 27 | 80 | 173 |
| 34 | 132 | 27 | 17 | 134 | 170 | 158 | 23 | 117 | 39 |
| 35 | 18 | 170 | 18 | 132 | 80 | 134 | 117 | 116 | 38 |
| 158 | 44 | 81 | 44 | 17 | 117 | 133 | 116 | 178 | 191 |

Cont. on next page.

| Day21 | Day22 | Day23 | Day24 | Day25 | Day26 | Day27 | Day28 | Day29 | Day30 |
|---|---|---|---|---|---|---|---|---|---|
| 134 | 19 | 117 | 176 | 18 | 116 | 132 | 115 | 34 | 23 |
| 17 | 79 | 116 | 54 | 19 | 115 | 18 | 158 | 158 | 117 |
| 44 | 176 | 178 | 174 | 79 | 177 | 44 | 134 | 134 | 116 |
| 43 | 55 | 34 | 135 | 176 | 158 | 176 | 132 | 132 | 115 |
| 176 | 54 | 158 | 145 | 103 | 132 | 25 | 17 | 17 | 178 |
| 25 | 103 | 132 | 144 | 70 | 17 | 55 | 18 | 19 | 34 |
| 103 | 70 | 44 | 109 | 160 | 44 | 54 | 43 | 176 | 159 |
| 70 | 161 | 176 | 148 | 175 | 43 | 160 | 79 | 55 | 158 |
| 161 | 174 | 54 | 21 | 135 | 79 | 174 | 54 | 54 | 134 |
| 174 | 175 | 103 | 90 | 143 | 103 | 135 | 103 | 70 | 132 |
| 175 | 135 | 161 | 163 | 21 | 4 | 143 | 70 | 161 | 17 |
| 135 | 145 | 160 | 85 | 91 | 160 | 91 | 161 | 175 | 18 |
| 145 | 143 | 174 | 14 | 186 | 174 | 186 | 174 | 135 | 44 |
| 144 | 109 | 175 | 76 | 122 | 175 | 163 | 175 | 145 | 43 |
| 109 | 148 | 135 | 180 | 85 | 135 | 122 | 145 | 109 | 19 |
| 148 | 91 | 145 | 29 | 15 | 145 | 86 | 144 | 21 | 79 |
| 21 | 90 | 109 | 140 | 162 | 144 | 14 | 109 | 91 | 176 |
| 90 | 163 | 32 | 71 | 180 | 148 | 15 | 90 | 90 | 128 |
| 163 | 122 | 21 | 45 | 71 | 21 | 111 | 186 | 186 | 174 |
| 122 | 14 | 90 | 190 | 167 | 91 | 162 | 163 | 89 | 135 |
| 120 | 15 | 163 | 124 | 45 | 90 | 76 | 86 | 163 | 145 |
| 14 | 162 | 122 | 110 | 137 | 186 | 75 | 85 | 126 | 52 |
| 15 | 76 | 120 | 164 | 36 | 163 | 139 | 120 | 14 | 91 |
| 162 | 75 | 14 | 60 | 59 | 126 | 71 | 14 | 162 | 90 |
| 75 | 73 | 15 | 61 | 60 | 85 | 187 | 162 | 180 | 186 |
| 74 | 138 | 162 |  | 61 | 15 | 45 | 76 | 75 | 163 |
| 139 | 68 | 77 |  |  | 162 | 189 | 180 | 74 | 126 |
| 140 | 190 | 76 |  |  | 76 | 110 | 138 | 73 | 120 |
| 71 | 110 | 180 |  |  | 180 | 36 | 190 | 138 | 162 |
| 68 | 164 | 53 |  |  | 75 | 59 | 137 | 29 | 180 |
| 189 | 60 | 75 |  |  | 74 | 61 | 164 | 71 | 138 |
| 190 | 61 | 140 |  |  | 138 |  | 166 | 187 | 139 |
| 36 |  | 45 |  |  | 71 |  | 59 | 45 | 140 |
| 166 |  | 68 |  |  | 167 |  | 60 | 68 | 71 |
| 61 |  | 190 |  |  | 188 |  | 61 | 190 | 72 |
|  |  | 124 |  |  | 190 |  |  | 110 | 45 |
|  |  | 164 |  |  | 124 |  |  | 36 | 68 |
|  |  | 59 |  |  | 164 |  |  | 166 | 190 |
|  |  | 60 |  |  | 59 |  |  | 61 | 36 |

Cont. on next page.

| Day21 | Day22 | Day23 | Day24 | Day25 | Day26 | Day27 | Day28 | Day29 | Day30 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 60 | | | | 164 |
| | | | | | 61 | | | | 166 |
| | | | | | | | | | 59 |
| | | | | | | | | | 60 |

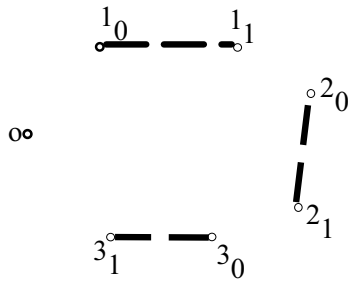
Table C.1: *Daily Realized Street Segments for Problem Set 1.*

# Appendix D

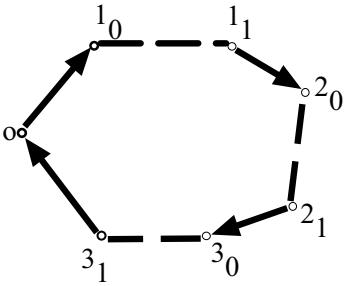
## Calculation of Expected Length

Consider a small example with three street segments that could require service as shown in Figure D.1 (a). The route starts and ends at the same location: the depot (node 0). We wish to compute the expected length for a given route $\tau = (0, arc1, arc2, arc3, 0)$ as illustrated in Figure D.1 (b). Let $p_i$ and $d(i_0, i_1)$ be the probability of requiring service and the length of street segment $i$, respectively. Also, let $d(i_1, j_0)$ denote the shortest-path road network distance between $i_1$ and $j_0$.

Observe that for an instance with $n$ street segments (arcs), a route $\tau$ has $2^n$ possible realizations: $T_m^\tau$, where $m = 1, 2, ...2^n - 1, 2^n$ and each realization represents a configuration determined by whether each of the n arcs requires service or not. Furthermore, we can easily calculate the probability for each realization $T_m^\tau$, $P(T_m^\tau)$, based on the probability of requiring service of the street segments. Let $L(T_m^\tau)$ be the length of realization $T_m^\tau$. Apparently, a straightforward but inefficient approach




(a) 0 is the depot. Three arcs are 1, 2 and 3.




(b) A route with the arc sequence 1 -> 2 -> 3.

Figure D.1: *An example with three arcs.*

to compute the expected length for a given $\tau$ is:

$$E[L(\tau)] = \sum_{m=1}^{2^n} P(T_m^\tau) \times L(T_m^\tau) \tag{D.1}$$

Here we use an efficient alternative based on the PTSP expected length calculation proposed by Bertsimas et al. (1990). In particular, we calculate $E[L(\tau)]$ as the summation of the following four components (see (5.1) to (5.4)):

1. The expected cost of having a direct arc from the depot to each street segment:

   $d(0, 1_0)p_1 + d(0, 2_0)p_2q_1 + d(0, 3_0)p_3q_1q_2$

2. The expected cost associated with having an arc between each pair of street segments: $d(1_1, 2_0)p_1p_2 + d(1_1, 3_0)p_1p_3q_2 + d(2_1, 3_0)p_2p_3$

3. The expected cost of having a direct arc from each street segment back to the

depot: $d(1_1, 0)p_1q_2q_3 + d(2_1, 0)p_2q_3 + d(3_1, 0)p_3$

4. The expected cost of using each street segment: $d(1_0, 1_1)p_1 + d(2_0, 2_1)p_2 + d(3_0, 3_1)p_3$

# Bibliography


[1] A.A. Assad and B.L. Golden, Arc routing methods and applications, *Handbooks in operations research and management science, vol 8*, M.O. Ball, T.L. Magnanti, C.L. Monma and G. L. Nemhauser (Editors), Amsterdam: Elsevier, 375-483, (1995).

[2] C. Archetti, A. Hertz, and M. Speranza, A tabu search algorithm for the split delivery vehicle routing problem, Transport Sci 40, 64-73, (2006).

[3] C. Archetti, Private communication, (2005).

[4] G.P. Agrawal, Nonlinear fiber optics, Academic Press, San Diego, CA, Chap. 1, (2001).

[5] Blue Gene. (n.d.), Retrieved on Oct 15, 2006, from *http* : *//www.research.ibm.com/bluegene*.

[6] A. Birman, Computing approximate blocking probablities for a class of all-optical networks, IEEE J Selected Areas Commun 14, 852-857, (1996).

[7] D. Bertsimas, P. Jaillet and A. Odoni, A priori optimization, Oper Res 38, 1019-1033, (1990).

[8] D. Bertsimas and L. Howell, Further results on the probabilistic traveling salesman problem, Euro J Oper Res 65, 68-95, (1993).

[9] J. Belenguer, M. Martinez, and E. Mota, A lower bound for the split delivery vehicle routing problem, Oper Res 48, 801-810, (2000).

[10] M.S. Borella, J.P. Jue, D. Banerjee, B. Ramamurthy, and B. Mukherjee, Optical components for WDM lightwave networks, Proc IEEE 85, 1274-1307, (1997).

[11] R.A. Barry and P.A. Humblet, Models of blocking probability in all-optical networks with and without wavelength changers, IEEE J Selected Areas Commun 14, 858-867, (1996).

[12] A. Campbell, Aggregation for the probabilistic traveling salesman problem, Comput Oper Res 33, 2703-2724, (2006).

[13] A. Campbell and B. Thomas, Probabilistic traveling salesman problem with deadlines, Transport Sci, under review.

[14] A. Campbell and B. Thomas, Challenges and advance in a priori routing, submitted to *The vehicle routing problem: Latest advances and challenges*, Bruce Golden, Raghu Raghavan, and Edward Wasil (Editors), Springer, August, (2007).

[15] A. Corberan, R. Mart and A. Romero, Heuristics for the mixed rural postman problem, Comput Oper Res 27, 183-203, (2000).

[16] G. Clarke and J. W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Oper Res 12, 568-581, (1964).

[17] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany, New heuristics for the vehicle routing problem, in *Logistics systems: Design and optimization*, A. Langevin and D. Riopel (Editors), Springer, 270-297, (2005).

[18] N. Christofides and S. Eilon, An algorithm for the vehicle-dispatching problem, Oper Res Quarterly 20, 309-318, (1969).

[19] N. Christofides, A. Mingozzi, and P. Toth, The vehicle routing problem, in *Combinatorial optimization*, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi (Editors), John Wiley & Sons, Chichester, UK, 315-338, (1979).

[20] M. Dror and P. Trudeau, Savings by split delivery routing, Transport Sci 23, 141-145, (1989).

[21] M. Dror and P. Trudeau, Split delivery routing, Naval Res Logist 37, 383-402, (1990).

[22] M. Dror, G. Laporte, and P. Trudeau, Vehicle routing with split deliveries, Discrete Appl Math 50, 239-254, (1994).

[23] H.A. Eiselt, M. Gendreau and G. Laporte, Arc routing problems, part II: The rural postman problem, Oper Res 43, 399-414, (1995).

[24] R. Elsenpeter and T.J. Velte, *Optical networking: A beginner's guide*, Osborne: McGraw-Hill, (2002).

[25] S. Eilon, C. Watson-Gandy, and N. Christofides, *Distribution management: Mathematical modeling and practical analysis*, Griffin, London, (1971).

[26] M.C. Fu, Optimization for simulation: Theory vs. practice, INFORMS J Comput 14, 192-215, (2002).

[27] P. Frizzell and J. Giffin, The bounded split delivery vehicle routing problem with grid network distances, Asia-Pacific J Oper Res 9, 101-116, (1992).

[28] P. Frizzell and J. Giffin, The split delivery vehicle scheduling problem with time windows and grid network distances, Comput Oper Res 22, 655-667, (1995).

[29] R. Frisch, *Maxima and minima: Theory and economic applications*, Chicago, IL: McNally, (1966).

[30] R.W. Floyd, Algorithm 97: Shortest path, Commun ACM archive 5, 345, (1962).

[31] L. Gouveia, P. Patricio, A. Sousa and R. Valadas, MPLS over WDM network design with packet level QoS constraints based on ILP modes, Proc IEEE Infocom 1, 576-586, (2003).

[32] M.R. Garey and D.S. Johnson, *Computers and intractability: A guide to the theory of NP-Completeness*, Freeman, NY, (1979).

[33] M. Gen and L. Lin, Multiobjective hybrid genetic algorithm for bicriteria network design problem, The 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems, December, (2004).

[34] T.J. Gaskell, Bases for vehicle fleet scheduling, Opl Res Quart 18, 281-295, (1967).

[35] K. Helsgaun, An effective implementation of the Lin-Kernighan travelling salesman heuristic, Euro J Oper Res 126 , 106-130, (2000).

[36] M.D. Intriligator, *Mathematical optimization and economic theory*, Englewood Cliffs, NJ: Prentice-Hall, (1971).

[37] C.S. Li, F.F.K. Tong, C.J. Georgiou and M. Chen, Gain equalization in metropolitan and wide area optical networks using optical amplifiers, Proc IEEE Infocom 1, 130-137, (1994).

[38] F. Li, B. Golden, and E. Wasil, Very large-scale vehicle routing: New test problems, algorithms, and results, Comput Oper Res 32, 1197-1212, (2005).

[39] G. Laporte, F. V. Louveaux and H. Mercure, A priori optimization of the probabilistic traveling salesman problem, Oper Res 42, 543-549, (1994).

[40] G. Laporte, Modeling and solving several classes of arc routing problems as traveling salesman problems, Comput Oper Res 24, 1057-1061, (1997).

[41] J. Linderoth, A. Shapiro and S. Wright, The empirical behavior of sampling methods for stochastic programming, Ann Oper Res 142, 215-241, (2006).

[42] K. Liu, A study on the split delivery vehicle routing problem, Ph.D. Dissertation, Department of Industrial Engineering, Mississippi State University, Mississippi State, Mississippi, (2005).

[43] L. Levy, Private communication, RouteSmart Technologies, Inc., (2006).

[44] R. Levi, R.O. Roundy and D.B. Shmoys, Provably near-optimal sampling-based policies for stochastic inventory control models, *38th Annual ACM Symposium on Theory of Computing*, (2006).

[45] Yu-Hsin Liu, A hybrid scatter search for the probablistic traveling salesman problem, Comput Oper Res 34, 2949-2963, (2007).

[46] P. Jaillet, The probabilistic travelling salesman problem, Technical report 185, Operations Research Center, MIT, Cambridge, Mass., (1985).

[47] P. Jaillet, A priori solution of a traveling salesman problem in which a random subset of the customers are visited, Oper Res 36, 929-936, (1988).

[48] A.J. Kleywegt, A. Shapiro and T. Homem-de-Mello, The sample average approximation method for stochastic discrete optimization, SIAM J. on Optimization 12, 479-502, (2002).

[49] A.S. Manne and A.F. Veinott,JR., Optimal plant size with arbitrary increasing time paths of demand, in *Investments for Capacity Expansion: Size, location and time phasing*, A. Manne (Editor), MIT Press, Cambridge, MA, Chapter 11, 178-190, (1967).

[50] B. Mukherjee, Optical communicaiton networks, New York: McGraw-Hill, (1997).

[51] B. Mukherjee, WDM optical communication networks: progress and challenges, IEEE J Selected Areas Commun 18, 1810-1824, (2000).

[52] P. Mullaseril, M. Dror, and J. Leung, Split-delivery routeing heuristics in livestock feed distribution, J Oper Res Society 48, 107-116, (1997).

[53] M.A. Nowak, The pickup and delivery problem with split loads, Ph.D. Thesis, Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, (2005).

[54] F. Ortega and L. Wolsey, A branch-and-cut algorithm for the single commodity uncapacitated fixed charge network flow problem, Retrived on Oct 15, 2006, from $http://ideas.repec.org/p/fth/louvco/0049.html$.

[55] J. Puchinger and G.R. Raidl, Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification, in *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, Spinger Berline/Heidelberg, 41-53, (2005).

[56] B. Ramamurthy, J. Iness, and B. Mukherjee, Optimization amplifier placements in a multi-wavelength optical LAN/MAN: The equallly-powered-wavelengths case, IEEE/OSA J Lightwave Technol 16, 1560-1569, (1998).

[57] B. Ramamurthy, J. Iness, and B. Mukherjee, Optimization amplifier placements in a multi-wavelength optical LAN/MAN: the unequallly-powered -wavelengths case, IEEE/OSA J Lightwave Technol 6, 755-767, (1998).

[58] B. Ramamurthy, H. Feng, D. Datta, J.P. Heritage and B. Mukherjee, Transparent vs. opaque vs. translucent wavelength-routed optical networks, Tech Dig Optical Fiber Commun(OFC '99), San Diego, CA, 59-61, (1999).

[59] R. Ramaswami and K. N. Sivarajan, Optical routing and wavelength assignment in all-optical networks , IEEE/ACM Trans Networks 3, 489-500, (1995).

[60] S. Raghavan, Michael O. Ball, and Vinai S. Trichur, Bicriteria Product Design Optimization: An Efficient Solution Procedure Using AND/OR Trees, Naval Res Logis 49, 574-592, (2002).

[61] A. Shapiro, Stochastic programming approach to optimization under uncertainty, Math Programming Ser. B 12, 38, (2007).

[62] A. Shapiro, and A. Nemirovski, On complexity of stochastic programming problems, *Continuous optimization*, Springer US, 111-146, (2005).

[63] G. Sierksma and G. Tijssen, Routing helicopters for crew exchanges on off-shore locations, Ann Oper Res 76, 261-286, (1998).

[64] Success Stories. (n.d.). Retrieved on Oct 15, 2006, from $http://www.science-ofbetter.org/cando/successstories/mepdu.htm$.

[65] S. Song, K. Lee, and G. Kim, A practical approach to solving a newspaper logistics problem using a digital map, Comput Ind Eng 43, 315-330, (2002).

[66] Transportation of the United States, Retrieved on Jan 25, 2007, from *http* : *// www.nationalatlas.gov/transportation.html*.

[67] B. Verweij, S. Ahmed, A.J. Kleywegt, G. Nemhauser and A. Shapiro, The sample average approximation method ppplied to stochastic routing problems: A computational study, Comput Optim Appl 24, 289-333, (2003).

[68] J. Valdes, R.E. Tarjan, and E.L. Lawler, The recongnition of series-parallel graphs, SIAM J Compute 11, 289-313, (1982).

[69] J. Ward, Minimum-aggregate-concave-cost multicommodity flows in strong-series-parallel networks, Math Oper Res 24, 106-129, (1999).

[70] R. Wong, Vehicle routing for small package delivery and pickup services, *The vehicle routing problem: Latest advances and challenges*, Bruce Golden, Raghu Raghavan, and Edward Wasil (Editors), Springer, August, (2007).

[71] E. Yetginer and E. Karasan, Regenerator placement and traffic engineering with restoration in GMPLS networks, Photonic Network Commun 6, 139-149, (2003).

[72] P. Yellow, A computational modification to the savings method of vehicle scheduling, Ops Res Quart 21, 281-283, (1970).

[73] YouTube Fact Sheet, Retrived on Oct 15, 2006, from *http* : *//www.youtube.com /t/factsheet*.

[74] A. Zymolka, Untersuchung eines Tourenplanungsproblems, Master Thesis, Mathematics, Philipps-University Marburg, Germany, (1999).