Université de Montréal

A Study on Two Arc Routing Problems

par

Srimathy Govindan

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en informatique

Mai, 1998

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée:

A Study on Two Arc Routing Problems

presentée par:

Srimathy Govindan

A été évaluée par un jury composé des personnes suivantes:

...Sang Nguyen.....:	président-rapporteur
...Michel Gendreau.....:	directeur de recherche
...Jean-Marc Rousseau.....:	codirecteur
...Jean-Yves Potvin.....:	membre du jury
...Richard Eglese.....:	examinateur externe

Thèse acceptée le: ...15.07.1998.....

ABSTRACT

This thesis focuses on two specific problems in arc routing – the *Capacitated Arc Routing Problem* (CARP) and the *Stochastic Eulerian Tour Problem* (SETP). Both problems have excellent application potential and deserve special attention.

The first problem that we consider is the CARP. We have developed a tabu search based heuristic procedure, TABUCARP, for this problem. In contrast to CARPET, another recent tabu search based heuristic for the CARP by Hertz et al. (1996), our heuristic considers a secondary objective of balancing the total work load (demand) fairly equally among the routes, in addition to minimizing the total cost. We feel that it is quite important to incorporate this feature, since most applications such as mail delivery, meter reading, and garbage collection require work load balancing.

TABUCARP continuously moves capacity excesses and deficits from routes farther away from the depot to one of the neighboring routes. We incorporate several features such as self-adjusting penalties, random tabu tags, and adaptive memory to guide the search. We have tested our algorithm on a set of 23 test problems by DeArmon, and another set of random problems. TABUCARP produces routes similar to CARPET for the DeArmon problems. A comparison of TABUCARP and CARPET solutions on the random problems indicates that it is definitely advantageous to consider work load balancing as an additional objective. The price we pay for gaining work load balance in the TABUCARP solutions is minimal since the deterioration in the objective function (total distance traveled) of the TABUCARP solutions is marginal when compared to the objective function of CARPET solutions.

The second problem that we consider in this thesis is the *Stochastic Eulerian Tour Problem* (SETP). The SETP arises when the set of edges that have to be visited (the set of present edges) on any particular day is random. The investigation of this

problem was motivated by a situation in the UK postal system. Here, the postal carriers deliver mail twice every day. During the morning delivery, typically all the streets in the network require service. However, during the afternoon delivery, the number of streets with demand is a very small subset of the set of all streets in the network and this subset usually varies from day to day. Given this scenario, the mail carrier, while following his morning route, usually skips the streets without demand during the afternoon delivery. Thus, given an undirected graph $G = (V, E)$ where all the edges in E require service, a distance matrix D and a probability distribution for the number of present edges, the SETP seeks an a priori Eulerian tour of minimum expected length.

We derive a closed form expression for the expected length of a given tour when the number of present edges follows a binomial distribution. We also show that the SETP is NP-complete, and derive further properties and a worst case ratio for the deviation of the expected length of a random Eulerian tour from the optimal tour in the expected sense. We also investigate some of the desirable properties of a good a priori tour using illustrative examples.

We have also developed three heuristics for the SETP. The first heuristic starts with an empty tour and determines the next edge to service as the one that results in the minimum expected increase in the length when appended at the end of the tour. The second heuristic selects the next edge of the tour from the set of edges at the current node rather than from the set of all available edges that requires service. Finally the third heuristic constructs several small sub-tours and then concatenates these sub-tours while considering the expected savings in concatenating sub-tours. We have also incorporated an adaptation of the US post-optimization procedure developed by Gendreau et al. (1992) for the TSP.

We have tested the performance of the three heuristics on grid networks and Euclidean graphs of various sizes. Our results indicate that when the probability of occurrence of the edges requiring service is very low ($p = 0.1$), the first heuristic seems to perform better than the other two heuristics. In other situations, the third heuristic seems to perform well for the grid networks. For the Euclidean networks, as the number of edges increases, the second heuristic seems to perform the best among

the three, though the margin of improvement is small. We also compared the expected lengths of the tours produced by the three heuristics with the expected length of a random Eulerian tour. Our results show that for the grid networks, the expected length of our best solution is lower than the expected length of a random tour by 10% on average, for lower values of p. As p increases to 1.0, this average reduces to 6% for $p = 0.5$ and 1% for $p = 0.9$. For the Euclidean networks, the expected length of our best solution is lower than the expected length of a random tour by 2% on average, for low values of p.

RÉSUMÉ

Un problème commun dans les systèmes de service est la conception des parcours pour les véhicules ou les personnes donnant ces services. Ces problèmes de tournées représentent un champ de recherche important depuis bon nombre d'années, ceci est dû, entre autres, à l'abondance d'applications. Avec l'augmentation du coût des véhicules et de la main d'œuvre, toute économie obtenue grâce à l'amélioration de la conception des parcours sera appréciée.

Les problèmes de tournées se divisent en deux classes – *Les problèmes de tournées sur les nœuds* (PTN) et *les problèmes de tournées sur les arcs* (PTA). Dans un PTN, la demande (collecte ou livraison) se produit sur les nœuds d'un graphe alors que dans un PTA, la demande se situe au niveau des arcs du graphe. Dans un PTA, l'objectif est de construire des routes à coût minimal qui traversent un ensemble donné d'arcs nécessitant un service. En dépit d'un nombre incroyable d'applications réelles, les PTA n'ont pas reçu autant d'attention de la part des chercheurs que les PTN. Alors que des progrès considérables ont été accomplis dans le développement d'excellents résultats théoriques et de procédures donnant des solutions exactes pour les PTN, la recherche faite du côté des PTA semble se limiter à des situations particulières. Avec cette motivation en tête, cette thèse se concentre sur deux problèmes précis de tournées sur les arcs -- Le *problème de tournées sur les arcs avec capacité* (PTAC) et le *problème de tournée stochastique eulérienne* (PTSE). Ces deux problèmes possèdent un excellent potentiel d'application et demandent une attention particulière.

Le premier problème que nous considérons est le PTAC. Il s'agit d'un des problèmes les plus importants de tournées sur des arcs, ceci étant dû à leurs présences dans des applications telles que le déneigement, le nettoyage des rues, la collecte des ordures, la distribution de courrier postal et plusieurs autres. Le PTAC est un

problème très difficile et il serait irréaliste de croire que des procédures exactes puissent être utilisées pour résoudre ce problème, même en se limitant à des tailles moyennes. Les chercheurs ont développé de nombreuses heuristiques pour les PTAC. La plupart de celles-ci constituent une première tentative et très peu de travail a été fait dans le développement de procédures d'amélioration locale. De plus, chaque heuristique semble plutôt adaptée à des types particuliers de graphes. Récemment, Hertz et al. (1996) ont développé une heuristique pour le PTAC qui est basée sur la recherche avec tabous (CARPET).

CARPET évolue d'une solution vers une solution voisine en déplaçant un arc à desservir de la tournée courante vers une autre tournée. Cette heuristique est semblable à TABUROUTE (l'heuristique de recherche avec tabous pour les problèmes de tournées de véhicules (Gendreau et al. 1994)) dans la plupart de ses caractéristiques. L'objectif de CARPET est de produire une solution au PTAC qui ait le coût le plus faible possible. Cependant, dans beaucoup de situations pratiques, en plus de vouloir minimiser le coût total, il y a un second objectif qui consiste à équilibrer la charge totale de travail de manière équitable pour chacune des routes. Nous tenons compte aussi de cet objectif dans notre heuristique et tentons de produire des solutions équilibrées à coût minimal.

Nous pensons qu'il est très important d'inclure cet aspect puisque la plupart des applications telles que la distribution du courrier, la lecture des compteurs électriques et la collecte des ordures demandent une charge de travail equilibrée. Dans un contexte réel, si l'algorithme utilisé ne considère que le coût total, le répartiteur doit généralement revoir la solution et la modifier à vue d'œil afin de l'équilibrer. À l'inverse, notre procédure (TABUCARP) vise à être plus globale et considère cette caractéristique dans l'algorithme. Ainsi, celui-ci peut servir d'outil de planification pour plusieurs problèmes de tournées.

TABUCARP déplace constamment les excès et déficits de capacité des chemins qui sont éloignés du dépôt vers une des routes voisines du dépôt. Le déplacement de base consiste à ajouter (enlever) un ou plusieurs arcs de service à une (d'une) tournée qui est sous (sur) utilisée. Nous avons utilisé plusieurs composantes telles que des pénalités auto-adaptables, des étiquettes tabous aléatoires et une

mémoire adaptative pour guider la recherche. Nous avons testé notre algorithme sur un ensemble de 23 problèmes tests de DeArmon ainsi que sur des problèmes générés aléatoirement. Nous avons comparé nos résultats avec ceux de CARPET. TABUCARP produit des trajets similaires à CARPET dans le cas des problèmes de DeArmon. Pour les problèmes aléatoires, les solutions de TABUCARP sont améliorées de 13,6 % en moyenne par rapport à celles de CARPET, en terme d'équilibre de la charge de travail.

Le deuxième problème que nous avons considéré dans la thèse est le *problème de tournée stochastique eulérienne* (PTSE). Pour le PTSE l'ensemble des arcs à visiter pour une journée particulière est aléatoire. L'étude de ce problème a été suscitée par une application réelle : dans le système postal anglais, un facteur peut distribuer le courrier une deuxième fois dans l'après-midi; le nombre de rues à visiter est alors très petit et varie d'une journée à l'autre. Dans un tel cas, le facteur, tout en suivant son trajet régulier, peut généralement sauter les rues qui ne demandent pas de visite.

Il est important de remarquer qu'il existe peut-être plus d'un parcours eulérien pour un graphe donné. Cependant tous ces parcours possèdent le même coût, et de par ce fait, aucune optimisation n'est nécessaire pour le problème de tournée déterministe eulérienne. Toutefois, pour le PTSE, chaque parcours présente certains avantages et inconvénients par rapport au saut d'arcs et a donc des longueurs espérées différentes. Donc, étant donné un graphe nonorienté $G = (V, E)$ où tous les arcs de E nécessitent un service, une matrice de distances D et une distribution de probabilité pour le nombre d'arcs requis, le PTSE cherche à determiner a priori un parcours eulérien ayant une longueur espérée minimale.

Le PTSE n'a pas été étudié dans la littérature jusqu'à maintenant. Nous pensons qu'il joue un rôle important dans les situations où le nombre d'arcs à visiter chaque jour est aléatoire et petit, comparé au nombre total d'arcs présents. Ceci motive notre examen de ce problème et de ses propriétés, ainsi que le développement d'algorithmes spécifiques pour le résoudre.

Nous avons déterminé une expression pour la longueur espérée d'un parcours donné lorsque le nombre d'arcs présents suit une distribution binomiale. Ce résultat

peut aisément être étendu au cas où le nombre d'arcs à desservir suit n'importe quelle loi de probabilité discrète. Nous montrons également que le PTSE est NP-complet, et ce, même si la contrepartie déterministe se résoud en temps polynomial. Nous présentons d'autres propriétés ainsi qu'une borne sur la déviation de la longueur espérée du parcours eulérien aléatoire relativement au parcours optimal. Nous nous penchons aussi sur les propriétés souhaitables d'un bon parcours a priori, ceci en utilisant des exemples explicatifs.

Puisque le PTSE appartient à une classe de problèmes difficiles, il n'est pas possible de résoudre des problèmes de taille réaliste en utilisant des algorithmes s'exécutant en temps polynomial. Nous nous concentrons donc sur le développement d'algorithmes heuristiques donnant de bonnes solutions. Nous avons construit trois heuristiques pour le PTSE. La première, une heuristique gloutonne globale, commence avec un parcours vide et choisit le prochain arc en déterminant celui qui ajoutera au parcours la longueur espérée minimale, lorsque l'arc est servi à la fin du parcours. La seconde, une heuristique gloutonne locale, choisit le prochain arc du parcours à partir de l'ensemble des arcs du nœud courant plutôt que de l'ensemble des arcs à desservir disponibles. Finalement, le troisième type, une heuristique de construction de sous- parcours, produit plusieurs petits sous-parcours et ensuite réunit ces sous-parcours en considérant les économies prévues dans un processus de concaténation. Nous avons aussi ajouté une adaptation de la procédure de post-optimisation US développée par Gendreau et al. (1992) pour le TSP.

Nous avons testé le rendement de ces trois heuristiques sur des réseaux quadrillés et des graphes eulériens de tailles variées. Nos résultats montrent que lorsque la probabilité de présence des arcs à desservir est petite, l'algorithme glouton global semble donner de meilleurs résultats que les deux autres. Dans d'autres situations, l'heuristique de construction de sous-parcours réagit bien pour des réseaux quadrillés. Pour les réseaux eulériens, lorsque le nombre d'arcs augmente, la deuxième heuristique donne le meilleur rendement, cependant l'amélioration est très faible. Puisque toutes les heuristiques sont assez rapides (les plus gros problèmes - des réseaux quadrillés de 9x9 et des réseaux eulériens de 20 nœuds avec densité 0,7

pour les arcs - prennent moins d'une minute pour chaque heuristique), nous pouvons utiliser les trois heuristiques et choisir la meilleure solution.

Nous comparons également les longueurs espérées des parcours produits par les trois heuristiques avec la longueur espérée d'un parcours eulérien aléatoire. Nos résultats montrent que pour les réseaux quadrillés, la longueur espérée de notre meilleure solution est inférieure à la longueur prévue d'un parcours aléatoire par 10% en moyenne, pour des petites valeurs de p. Lorsque p augmente, cette moyenne diminue à 6% pour $p = 0{,}5$ et à 1% pour $p = 0{,}9$. Dans un cas particulier, la longueur prévue du parcours produit par l'heuristique de construction de sous-parcours est de 25% inférieure à la longueur prévue du parcours aléatoire. Les écarts pour les réseaux eulériens ne sont pas aussi importants. La longueur prévue de notre meilleure solution est inférieure à la longueur prévue du parcours aléatoire par 2% en moyenne, pour de petites valeurs de p.

Cette thèse s'est penchée sur deux problèmes importants et intéressants dans le domaine des itinéraires sur des arcs. Nous avons développé une heuristique de recherche avec tabous pour le PTAC ayant comme objectif secondaire une charge de travail equilibrée pour chacun des parcours. Nous avons aussi introduit le PTSE et analysé plusieurs propriétés théoriques de ce problème. Finalement, nous avons construit trois heuristiques rapides pour le PTSE. Les recherches futures incluent l'examen de méthodes exactes et de nouvelles procédures d'amélioration pour le PTSE.

To Shirdi Sai Baba

ACKNOWLEDGEMENTS

My first note of appreciation and thanks goes to my three year old daughter, Prithvi, who understood that her mommy was doing her thesis and rarely disturbed me. She even stayed with her grandparents in India for a few months, when I was finishing the major portion of my thesis. She is extremely glad that I will be finished pretty soon and has a long list of things to do with me!

I would like to thank my thesis director, Prof. Michel Gendreau, and the co-director, Prof. Jean-Marc Rousseau, for their belief in my work and abilities. Prof. Gendreau has provided valuable insights and his expert comments during various stages of my thesis. He has been more than a thesis director. In spite of his busy schedule, he always found time to provide excellent advice about life in academia and possibilities for my career. He is a kind and considerate person and never forgets to enquire about my family. I would like to thank Prof. Rousseau for introducing me to the area of arc routing and providing the motivation for investigating the Stochastic Eulerian Tour Problem.

I would also like to thank the members of my thesis committee, Prof. Richard Eglese, Prof. Jacques Ferland, and Prof. Jean-Yves Potvin, for their constructive and timely feedback. A special note of thanks to Prof. Gilbert Laporte who represented the external examiner during my defense. He provided a lot of feedback and suggested several changes to make the thesis more complete. Thanks to Renée Touzin for help with the French translation and René Séguin for reading the French summary and making sure that it conveyed the main ideas and contributions of my thesis properly. I would like to thank Madame Claire Valois at the Département d'informatique et de recherche opérationnelle and the staff at the Centre de recherche sur les transports for helping me with the administrative requirements and making my stay at the Université de Montréal a very pleasant one.

This thesis would not have been possible but for the constant support and encouragement from my husband, my family, and my husband's family. I am very grateful to my husband for his patience and help all through my student life. He has been there for me, both during good and bad times. My father has constantly encouraged me to pursue my interests and taught me the value of honesty and integrity through his work ethics. My mother has been a great friend and companion all along and is surely the mother I would like to emulate in life. My parents-in-law have always supported my efforts and I very much appreciate their love and care.

My brother, sister, brother-in-law, and nephew Rolly have always added fun to my life and made me want to visit them as often as possible. Thanks to our beloved Sai Maa for all her spiritual guidance and blessings. Last but not the least, thanks to our second baby who is due in August, for making this pregnancy an easy one.

Finally, I would like to acknowledge the scholarships from FCAR and the Université de Montréal. I would also like to thank Michel Mittaz of École Polytechnique Fédérale de Lausanne for providing me with a copy of his code for CARPET and LB2, which helped me immensely during the computational testing of our algorithm for the CARP.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM MOTIVATION

A very common problem in service systems is the design of routes for vehicles or people delivering service. These *routing problems* have been an important area of research for a long time due to the abundance of practical applications. With the rising cost of labor and operating vehicles, any savings obtained through the design of better routes would be well appreciated. As a result, government and private organizations continue to encourage research on designing optimal or near-optimal routes for service delivery systems.

Routing problems fall into two classes - *Node Routing Problems* (NRPs) and *Arc Routing Problems* (ARPs). In an NRP demand (pickup or delivery) occurs on the nodes or vertices of a graph, while in an ARP the demand occurs along the arcs or edges of a graph. Typically, in an NRP the objective is to visit, at the lowest cost possible, a given set of points in order to satisfy the demands at these points. In an ARP, the objective is to design minimum cost routes that traverse a given set of arcs or edges that require service. The well-known *Traveling Salesman Problem* (TSP) belongs to the class of NRPs, and the *Chinese Postman Problem* (CPP) and the *Rural Postman Problem* (RPP) belong to the class of ARPs. Specific node routing examples include coin collection from public telephone booths, mail pickup from specific drop-off points, and distribution of newspapers to newsstands. Everyday problems of street sweeping, snow plowing, meter reading, school bus routing and household garbage collection are excellent examples of ARPs.

In spite of the numerous real world applications, ARPs have not received as much attention from researchers as NRPs. While researchers have made considerable progress in developing excellent theoretical results and exact solution procedures for several NRPs, the research work in the field of arc routing tends to be tailored to specific situations. With that as the motivation, this research focuses on two specific problems in arc routing – the *Capacitated Arc Routing Problem* (CARP) and the *Stochastic Eulerian Tour Problem* (SETP). Both problems have excellent application potential and deserve special attention. Before describing the objectives and contributions of this research in detail, we provide a review of the relevant literature on arc and node routing.

1.2 LITERATURE REVIEW

Researchers have been working on several theoretical and practical ARPs for a long time. The earliest work that we know of is the seven bridge problem at the Russian city of Königsberg (now Kaliningrad), examined by the great Swiss mathematician Leohnard Euler. It was in connection with a promenade to be taken across seven bridges that connected two islands with each other and with the two banks of the Pregel river (See Figure 1).

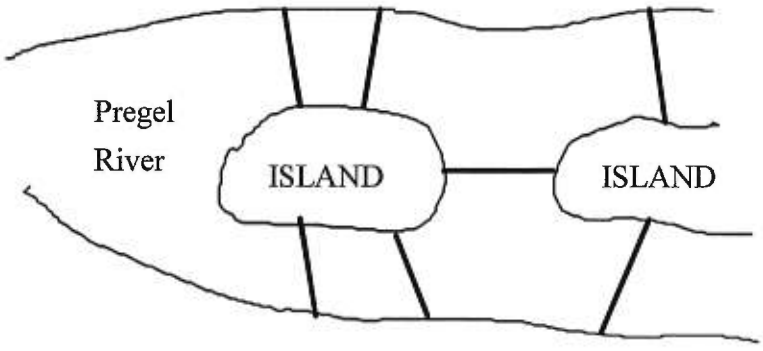

Figure 1. The seven bridges of Königsberg problem

Euler was interested in determining if there existed a way in which the promenade could cross the seven bridges exactly once. In 1736, Euler proved that this was not possible for this problem and derived the conditions for the existence of a closed walk on a graph G, containing each arc exactly once. A graph with this property is called an Eulerian or unicursal graph. Fleischner (1990) notes that in 1873, Heirholzer solved the problem of determining this closed walk on an undirected graph.

Guan (1962) documented the first ARP as we define it today. Guan defined the Chinese Postman Problem (CPP) as follows: "A mailman has to cover his assigned segment before returning to the post office. The problem is to find the shortest walking distance for the mailman". Guan showed that if a graph is not unicursal, it contains an even number of odd degree vertices and the graph can be made unicursal by replicating edges between odd degree vertices. He also derived the necessary and sufficient conditions for an Eulerian tour to be optimal.

Since the introduction of the CPP, several researchers have worked on the undirected and directed CPP and have shown that polynomial time algorithms exist for these problems. When only a subset of the links (edges or arcs) of the graph require service, the problem becomes the Rural Postman Problem (RPP). Lenstra and Rinnooy Kan (1976) proved that the undirected and directed RPP are NP-hard. Golden and Wong (1981) introduced the capacitated ARPs, and have shown that these problems are NP-hard. Hence, research has progressed in developing efficient exact procedures and heuristics for these hard problems. We discuss the nature of these research efforts in the following sections. A recent two-part survey by Eiselt, Gendreau, and Laporte (1995) and another survey by Assad and Golden (1995) present an overview of most of the relevant literature on arc routing. An earlier survey by Bodin et al. (1983) also provides information on arc routing problems.

1.2.1 UNCAPACITATED ARC ROUTING PROBLEMS

In the uncapacitated ARPs, the service delivery unit has unlimited capacity and hence, the single unit can serve the demand on all arcs. Polynomial time algorithms exist for

all uncapacitated ARPs, if the underlying graph G is Eulerian or unicursal. When G is not Eulerian, we need to augment G with a set of arcs or edges at the minimum cost to make it unicursal. For the undirected and the directed CPP, polynomial time algorithms exist for solving the least cost augmentation problem. All the other uncapacitated ARPs are NP-hard and the general trend is to use heuristics to determine a low cost augmentation of the graph. We review the relevant literature on the undirected, directed, and mixed versions of the CPP and the RPP below.

1.2.1.1 Undirected Chinese Postman Problem

We noted earlier that Euler first proved the necessary and sufficient condition for an undirected graph to be unicursal. Guan (1962) showed that a graph that is not unicursal always has an even number of odd degree vertices. This observation helped researchers (Edmonds 1965a, Busacker and Saaty 1965) to address the least cost augmentation problem as a weighted matching problem on the odd degree vertices of the underlying graph G.

We present below the integer linear programming formulation of the undirected CPP. The problem is to make the given graph $G=(V,E)$ a unicursal graph G' by replicating edges at the minimum cost. Let $x_{ij}\ (i<j)$ be the number of times edge (v_i,v_j) needs to be added to graph G, c_{ij} the cost of traversing edge (v_i,v_j), and $\delta(i)$ be the set of edges incident to vertex v_i. Also let $T\subseteq V$ be the set of odd degree vertices. Then the formulation is as follows.

$$\text{minimize} \quad \sum_{(v_i,v_j)\in E} c_{ij}x_{ij} \tag{1.1}$$

$$\text{subject to:} \sum_{(v_i,v_j)\in\delta(i)} x_{ij} \equiv \begin{cases} 1 \pmod 2 \text{ if } v_i \in T \\ 0 \pmod 2 \text{ if } v_i \in V\setminus T \end{cases} \tag{1.2}$$

$$x_{ij} \geq 0 \qquad \forall\ (v_i,v_j)\in E \tag{1.3}$$

$$x_{ij} \text{ integer} \qquad \forall (v_i,v_j)\in E \tag{1.4}$$

Constraints (1.2) ensure that the degree of all vertices in the graph are even and the objective function ensures that the set of edges added are at the lowest possible cost. As noted above, we can solve this problem as a minimum weighted matching problem on a graph $G_o = (T, E_T)$, where E_T is the set of edges connecting all the odd degree vertices, and the cost of each edge (v_i, v_j) in G_o is the cost of the shortest path between vertices v_i and v_j in G. Lawler (1976) provides an $O(|V|^3)$ time algorithm for solving the weighted matching problem. Once we solve the matching problem, we can augment G with the shortest paths corresponding to the optimal matching solution to obtain the unicursal graph G'.

Edmonds and Johnson (1973) have completely characterized the convex hull of (1.2), (1.3), and (1.4) as a polyhedron using additional inequalities called the blossom inequalities. The polyhedron given by (1.3) and the blossom inequalities completely describes the convex hull of the undirected CPP. Edmonds and Johnson provide an adaptation of Edmonds' (1965b) blossom algorithm for matching to solve this problem. Thus the least cost augmentation problem is well solved for the undirected CPP.

Edmonds and Johnson have also given three algorithms for determining an Eulerian cycle on the unicursal graph G'. Prior to this, a very simple algorithm due to Fleury was used. This algorithm starts at an arbitrary vertex v_i and traverses successively the edges of the graph while deleting each edge from G as it is traversed. At every point, care is taken not to traverse an edge whose removal would disconnect the remaining graph. While this method is very easy to follow, it is not well suited for computer solution since determining the next edge to delete at each step could be time consuming.

The three algorithms by Edmonds and Johnson are the End-Pairing algorithm, Next-Node algorithm, and the Maze-Search algorithm. All three algorithms are similar in that they begin by tracing out a simple tour which may not include all edges, form another non-overlapping tour starting at any node on the current tour, and then append the two tours to form a single longer tour. The procedure continues until

all edges are covered. The difference between the algorithms arises in the way the tours are appended. We present below a summary of the end-pairing algorithm.

End-Pairing Algorithm

Step 1: Trace out a simple tour. If it includes all edges, stop. If not, go to Step 2.

Step 2: Begin at any node v_0 on the tour incident to edges not on the tour and trace out another tour not overlapping the previous one.

Step 3: Combine the two tours into a single longer tour as follows. Let edges e_1 and e_2 be incident to node v_0 on the first tour, and edges e_3 and e_4 be incident to v_0 on the second tour. Start with edge e_3, traverse the second tour completely, and reach v_0 through edge e_4. Now follow the first tour starting with edge e_2 and reach v_0 through edge e_1. If all edges of the graph have been traversed, stop. Otherwise go to Step 2.

The following figure helps understand this concatenation process used in the end-pairing algorithm.

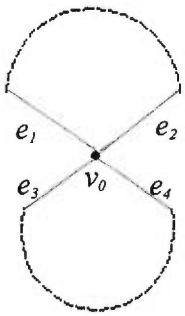

Figure 2. Concatenation of tours in the End-Pairing algorithm

1.2.1.2 Directed Chinese Postman Problem

The necessary and sufficient condition for a connected, directed graph G to be unicursal is that the graph be symmetric, i.e., the number of arcs entering and leaving each vertex must be equal. Note however that the graph has to be strongly connected for a solution to exist. When the graph is not symmetric, the additional arcs to be added to the graph to make it unicursal can be determined from the solution of a transportation problem (Edmonds and Johnson 1973, Beltrami and Bodin 1974, and Orloff 1974). The transportation problem is defined on the subgraph induced by the nodes for which the number of arcs entering and leaving the node are not equal, and all the arcs incident to such nodes. If the number of incoming arcs exceeds the number of outgoing arcs at a node v_i by s_i, then we can interpret s_i to be a supply at node v_i. Similarly, a node v_j for which the number of outgoing arcs exceeds the number of incoming arcs by d_j can be thought of as having a demand of d_j. Let I be the set of supply nodes and J, the set of demand nodes, and c_{ij} be the cost of the shortest path from a supply node v_i to a demand node v_j. The decision variable x_{ij} is 1 if we need to add a copy of all arcs corresponding to the shortest path from v_i to v_j, and 0 otherwise. The transportation problem can be formulated as follows.

$$\text{minimize} \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{1.5}$$

$$\text{subject to} \quad \sum_{v_j \in J} x_{ij} = s_i \qquad \forall\ v_i \in I \tag{1.6}$$

$$\sum_{v_i \in I} x_{ij} = d_j \qquad \forall\ v_j \in J \tag{1.7}$$

$$x_{ij} \geq 0 \qquad \forall\ v_i \in I, v_j \in J \tag{1.8}$$

The solution to this problem indicates the number of times each arc has to be replicated in the graph G to make it unicursal. Orloff (1974) notes that the least cost augmentation problem for the directed CPP can also solved as a minimum cost

network flow problem. The objective is to attain a minimum cost circulation on the network such that each arc flow is at least 1.

In order to determine the Eulerian tour, we can adapt Fleury's algorithm to a directed graph, but we still have the problem of choosing the next arc to traverse. Edmonds and Johnson (1973) present an algorithm by van Aardenne-Ehrenfest and de Bruijn (1951) to determine the tour. We now provide a step-by-step description of this algorithm.

van Aardenne-Ehrenfest and de Bruijn Algorithm

Step 1: Construct a spanning arborescence rooted at any vertex v_r.

Step 2: Label all arcs as follows: order and label the arcs outgoing from v_r in an arbitrary fashion; order and label the arcs out of any other vertex consecutively in an arbitrary fashion, so long as the last arc is an arc used in the arborescence.

Step 3: Obtain an Euler tour by first following the lowest labeled arc emanating from an arbitrary vertex; whenever a vertex is entered, it is left through the arc not yet traversed having the lowest label. The procedure ends with an Euler circuit when all arcs have been covered.

1.2.1.3 Mixed Chinese Postman Problem

The mixed CPP is defined on a graph G that contains both arcs and edges. Papadimitriou (1976) has shown that this problem is NP-hard even if the underlying graph is planar or if all c_{ij}s are equal. Hence, research has concentrated on developing efficient exact algorithms and heuristics that replicate enough arcs and edges in the graph to satisfy the necessary and sufficient conditions for unicursality. Once the graph is unicursal, we need to determine an Eulerian tour on it.

A mixed connected graph $G=(V, A \cup E)$ where A is the set of arcs and E is the set of edges, is said to be unicursal if and only if:

(i) every node is incident to an even number of directed and undirected arcs; and

(ii) for every $S \subseteq V$, the difference between the number of directed arcs from S to $V \setminus S$ and the number of directed arcs from $V \setminus S$ to S is less than or equal to the number of undirected arcs joining S and $V \setminus S$.

The second condition is called the balanced set condition and a graph that satisfies this property is called a balanced graph. If a graph is even and symmetric, then it is balanced. Several researchers have made use of this and have tried to make the underlying graph even and symmetric and thus, Eulerian. (See Edmonds and Johnson 1973, Frederickson 1979, Kappauf and Koehler 1979, Christofides, Benavent, Campos, Corberán, and Mota 1984, and Grötschel and Win 1992.) Nobert and Picard (1996) make the graph Eulerian by addressing the evenness and balanced set conditions directly. We present below a brief summary of the work directed at making a mixed graph Eulerian.

Exact Methods

(A) Kappauf and Koehler (1979) were the first to suggest an exact procedure for solving the mixed CPP. They have formulated the problem as an integer linear program and outlined an algorithm based on the exhaustive analysis of the extreme points of the underlying polyhedron. The formulation uses two integer variables for each edge and arc of the graph, for each direction of traversal. The constraints in the formulation ensure that each edge and arc is traversed at least once and force every vertex to be symmetric.

Grötschel and Win (1992) have used the same formulation along with additional valid inequalities (odd cut inequalities). They have devised a branch and cut procedure to solve instances of the mixed CPP. They use results from Minieka's research and adopt a three-way branching scheme. They have attempted 9 problems and solved all of them to optimality without any branching. The problem ranges were $52 \le |V| \le 172$, $37 \le |E| \le 154$, and $31 \le |A| \le 116$.

(B) Christofides et al. (1984) have used a similar formulation along with redundant constraints which state that the total arcs and edges incident to every vertex must be

even. They solve the mixed CPP using a branch and bound algorithm. They use two lower bounds obtained from two different Lagrangean relaxations of the problem formulation. The first relaxes the symmetry constraints. They solve the sub-problems as minimum cost perfect matching problems on the odd vertices of the graph. The second bound does not use the redundant even degree constraints. The authors relax the constraints that ensure that every edge and arc is traversed at least once, and solve a minimum cost flow problem to calculate the second bound. Using their enumerative algorithm, the authors have solved 34 randomly generated problems with $7 \le |V| \le 50$, $4 \le |E| \le 39$, and $3 \le |A| \le 85$.

(C) Nobert and Picard (1996) use an integer linear programming formulation which forces the even degree and balanced set conditions directly. Their formulation uses only one variable for each edge as opposed to the previous formulation. The authors also use a generalized form of Edmonds and Johnson's (1973) blossom inequalities to tighten the linear relaxation.

Nobert and Picard's algorithm starts with a linear program which includes the blossom inequalities associated with the odd nodes, the balanced set constraints associated with the unbalanced nodes and most unbalanced set of nodes, and the non-negativity constraints. The algorithm successively adds constraints corresponding to most unbalanced sets and to violated blossom inequalities. The authors check if the intermediate solution satisfies the balanced set condition by solving a maximum flow problem based on the previous work of Picard and Ratliff (1975). Picard and Queyranne's (1980) work helps to find all the most unbalanced sets if the graph is not balanced. At every iteration, balanced set constraints and blossom inequalities that are violated are added. If the solution is fractional and no more violated constraints can be added, Gomory cuts are added to help achieve integrality. Finally, branching can be used to gain integrality, if necessary. The algorithm has solved 313 problems out of 440 to optimality at the root of the tree. The problem ranges were $6 \le |V| \le 225$, $5 \le |E| \le 4455$, and $2 \le |A| \le 5569$.

Heuristic Methods

(A) Edmonds and Johnson (1973) present a heuristic algorithm for the mixed CPP. Frederickson (1979) calls it heuristic MIXED1 and shows that the algorithm has a worst-case bound of 2. He also modifies the algorithm to ensure that the graph is even at the end of the process. The algorithm first modifies the given graph to make the degree of each vertex even. This is done by solving a minimum cost matching problem on the odd vertices. The links used in the matching are added to G. The algorithm then makes the graph symmetric by solving a minimum cost flow problem. Edmonds and Johnson's version of the algorithm terminates at this stage showing the existence of a minimum cost solution to the network flow that maintains the even degree of each vertex. However, Frederickson shows that the network flow constraints are not sufficient to find such a solution. So he adds a third step to the algorithm to make sure that the graph is even, while keeping the graph symmetric.

(B) Frederickson (1979) presents another heuristic MIXED2, which is essentially the reverse of MIXED1. This algorithm first makes the graph symmetric using the same procedure as in the second step of MIXED1. In order to make the graph even, the algorithm then performs a minimum cost matching on the odd vertices of the augmented graph using shortest path distances between these vertices. The edges used in the matching are then added to the graph. Christofides et al. (1984) describe a procedure which can be considered equivalent to MIXED2. They suggest a simple improvement procedure also. For each arc (v_i, v_j), if c_{ij} is greater than the cost of a directed path from v_i to v_j, then arc (v_i, v_j) is deleted and the arcs in the shortest path are duplicated. This applies to edges also since they can be considered as two directed arcs. The authors provide some computational results also. On 34 test problems with $7 \le |V| \le 50$, $4 \le |E| \le 39$, and $3 \le |A| \le 85$, the procedure produced solutions within 3% of optimality on average.

Determining an Eulerian Tour

Once the mixed graph is made Eulerian using either an exact or a heuristic method, one has to determine the actual Eulerian tour. The general idea is to orient all the edges of the graph to make it completely directed and then use the van Aardenne-Ehrenfest and de Bruijn algorithm for directed graphs to determine the tour. If we know that the Eulerian graph is symmetric then the following simple procedure presented in Eiselt, Gendreau, and Laporte (1995), can be used to orient the graph.

Step 1: If all the edges are directed, stop.

Step 2: Let v be a vertex with at least one incident undirected edge (v,w). Set $v_1 \leftarrow v$ and $v_2 \leftarrow w$.

Step 3: Orient (v_1,v_2) from v_1 to v_2. If $v_2 = v$, go to Step 1.

Step 4: Set $v_1 \leftarrow v_2$ and identify an edge (v_1,v_2) incident to v_1. Go to Step 3.

Most exact and heuristic methods make the mixed graph Eulerian by making it even and symmetric. So the above procedure is sufficient to completely orient the graph. However some methods (Nobert and Picard 1991) make the graph Eulerian by making it even and balanced. It is important to note that a unicursal graph need not be symmetric. Hence, it is necessary to make the graph symmetric first and then use the above procedure.

Ford and Fulkerson (1962) have described a procedure to assign directions to some of the edges of a graph to make it symmetric. The procedure begins with replacing each edge in the graph with two oppositely directed arcs. All the arcs from the original graph are assigned a lower bound of 1 and the arcs obtained from the edges, a lower bound of 0. Also, all arcs are assigned an upper bound of 1. The authors suggest obtaining a feasible circulation on the new graph and using that solution to orient some of the edges as follows. If $(v_i,v_j) \in E$, $x_{ij} = 1$ and $x_{ji} = 0$, then orient the edge (v_i,v_j) from v_i to v_j. The resulting graph is symmetric.

Minieka (1979) introduced the *Windy Postman Problem* (WPP), a problem closely related to the mixed CPP. The problem is described on an undirected graph, but the cost of traversing the edges of the graph depends on the direction of travel. The objective of the WPP is to determine a least cost traversal of all the edges. Brucker (1981) and later Guan (1984) have shown that the WPP is NP-hard. Win (1989) proves that the WPP is solvable in polynomial time if the underlying graph is Eulerian. He also shows that the mixed CPP can be transformed into a WPP by properly defining the costs. Thus, the undirected, directed, and the mixed CPP can be thought of as special cases of the WPP. Grötschel and Win (1992) have used a formulation similar to the formulation of Kappauf and Koehler (1979) and have devised a cutting plane algorithm. They have solved 31 instances of the problem out of 36 with $52 \le |V| \le 264$, and $78 \le |E| \le 489$.

1.2.1.4 Undirected Rural Postman Problem

In the undirected RPP, the underlying graph $G = (V, E)$ is completely undirected. Associated with each edge is a cost c_{ij} of traversing it. The undirected RPP seeks a minimum cost traversal of a subset $R \subseteq E$ of the edges. The traversal may include edges from $E \setminus R$ if necessary. Note that when $R = E$, the undirected RPP reduces to an undirected CPP. Lenstra and Rinnooy Kan (1976) have shown that the undirected RPP is NP-hard. Hence, it is necessary to resort to heuristics to obtain a low cost augmentation for the undirected RPP. However, exact methods have also been developed.

All uncapacitated RPPs are solved on a modified graph $G' = (V', E')$. The purpose of this modification is to eliminate the unnecessary vertices from V and the unnecessary edges from $E \setminus R$. As a first step in this transformation, we form an intermediate graph $\overline{G} = (\overline{V}, \overline{E})$ as follows. Let $\overline{V} \subseteq V$ be the set of vertices that are incident to at least one edge in R and $\overline{E}$ be the set of edges in R and an edge (v_i, v_j) for every $v_i, v_j \in \overline{V}$ whose cost c_{ij} is the length of the shortest chain between v_i and

v_j. Finally, we get the modified graph $G' = (V', E')$ as follows. Set $V' = \overline{V}$, $E' = \overline{E}$, and delete all edges $(v_i, v_j) \in E' \setminus R$ for which $c_{ij} = c_{ik} + c_{kj}$ for some $k \in V'$, and all edges in $E' \setminus R$ which are one of two parallel edges if both edges have the same cost. This procedure can be applied to directed and mixed graphs also.

We illustrate this graph modification using the following simple example. Figure 3 shows the given undirected graph G where the plain arcs belong to the set of required arcs, the dashed arcs belong to the set $E \setminus R$, and the numbers along the edges denote the edge costs. The given graph contains seven vertices, four required edges and seven edges that belong to the set $E \setminus R$. Figure 4 shows the intermediate graph $\overline{G}$ and Figure 5 shows the transformed graph G' that contains five vertices, four required edges and three edges that belong to the set $E \setminus R$. Note that in the transformed graph G' the set R induces p connected components $G_1, \ldots, G_p$ with respective vertex sets $V_1, \ldots, V_p$ forming a partition of V'.

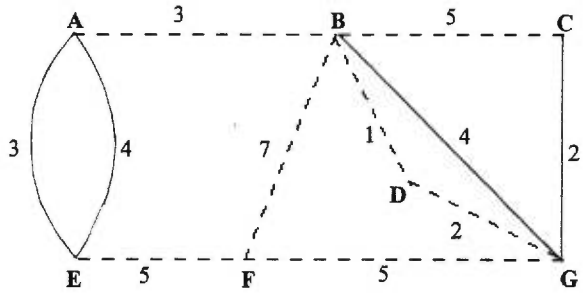

Figure 3. Original graph G

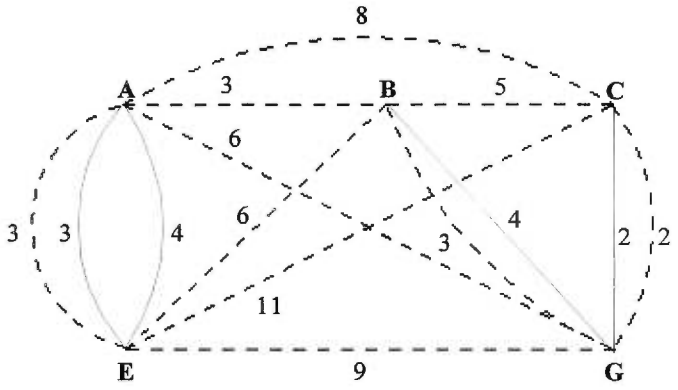

Figure 4. Intermediate graph $\overline{G}$

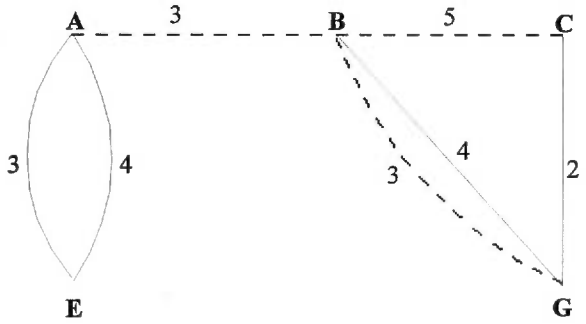

Figure 5. Transformed graph G'

Christofides, Campos, Corberán, and Mota (1981) have provided an integer linear programming formulation for the undirected RPP and have developed a branch and bound algorithm. In the formulation, the decision variables $x_{ij}\ (i < j)$ represent the number of times edge (v_i, v_j) is replicated in the optimal solution if $(v_i, v_j) \in R$.

If $(v_i, v_j) \in E' \setminus R$, then x_{ij} is the number of times edge (v_i, v_j) is traversed. The problem is formulated as follows.

$$\text{minimize} \quad \sum_{(v_i,v_j)\in R} c_{ij}(1+x_{ij}) + \sum_{(v_i,v_j)\in E''\setminus R} c_{ij}x_{ij} \tag{1.9}$$

$$\text{subject to} \quad \sum_{\substack{(v_i,v_j)\in R \\ j>i}} (1+x_{ij}) + \sum_{\substack{(v_i,v_j)\in R \\ j<i}} (1+x_{ji}) + \sum_{\substack{(v_i,v_j)\in E''\setminus R \\ j>i}} x_{ij} + \sum_{\substack{(v_i,v_j)\in E''\setminus R \\ j<i}} x_{ji} = 2z_i \qquad \forall\ v_i \in V' \tag{1.10}$$

$$\sum_{v_i\in S, v_j\in \bar{S}} x_{ij} \geq 1 \qquad \forall\ S = \bigcup_{k\in P} V_k, P \subset \{1,\ldots,p\} \tag{1.11}$$

$$x_{ij} \geq 0 \text{ and integer} \qquad \forall\ (v_i, v_j) \in E' \tag{1.12}$$

$$z_i \geq 0 \text{ and integer} \qquad \forall\ v_i \in V' \tag{1.13}$$

Constraints (1.10) force the degree of all the vertices in V' to be even and constraints (1.11) ensure that the p components of the graph G' are connected. The branch and bound solution procedure is very similar to that proposed by Held and Karp (1971) for the TSP. By relaxing constraints (1.10) in a Lagrangean fashion, it is easy to see that the problem reduces to a *Shortest Spanning Tree* (SST) problem on a graph whose vertices correspond to the connected components of G'. The authors use this relaxation to obtain a lower bound for the problem. In order to obtain an upper bound, the authors describe the following heuristic algorithm.

The algorithm first determines a SST solution T to connect the p components of G'. If all the vertices in the graph induced by $R \cup T$ are of even degree, then this graph is a solution to the augmentation problem. If the degree of some of the vertices is odd, then the authors determine a minimum cost matching on the odd vertices of this graph. Let M be the set of edges in the matching. Then the graph induced by $R \cup T \cup M$ is unicursal and an Eulerian tour can be determined on this graph. This heuristic provides an upper bound for the branch and bound algorithm. The authors have solved 24 randomly generated problems with $9 \leq |V| \leq 84$, $13 \leq |E| \leq 184$,

$4 \le |R| \le 78$, and $2 \le |p| \le 8$ to optimality. It is interesting to note that when the graph $G = (V, R)$ is connected, i.e., the graph G' has only one component, the problem reduces to a CPP on the graph induced by the set of edges in R.

1.2.1.5 Directed Rural Postman Problem

The directed RPP is defined on a completely directed graph $G = (V, A)$. As explained in the previous section, this problem is also solved on a transformed graph $G' = (V', A')$. Christofides, Campos, Corberán, and Mota (1986) have proposed a branch and bound algorithm and a heuristic for the directed RPP. Both are quite similar to the procedures described for the undirected RPP.

The integer linear programming formulation is as follows. The decision variables x_{ij} represent the number of times arc (v_i, v_j) is replicated in the optimal solution if $(v_i, v_j) \in R$. If $(v_i, v_j) \in A' \setminus R$, then x_{ij} is the number of times arc (v_i, v_j) is traversed. They use the following proposition to split the x_{ij} variables for $(v_i, v_j) \in R$. In graph G', for all $v_i \in V'$ with outdegree equal to 1 (indegree = 1) the arc $(v_i, v_j) \in R$ (arc $(v_j, v_i) \in R$) must be repeated at least r_{ij} = [indegree - 1] ([outdegree - 1]) times in any solution to the directed RPP. Thus, $x_{ij} = x'_{ij} + r_{ij}$. In order to make the exposition simple, we present a simplified formulation with the original x_{ij} variables, without loss of generality. The formulation with respect to a particular vertex set $\overline{V} \in \{V_1, \ldots, V_p\}$ is

$$\text{minimize} \quad \sum_{(v_i, v_j) \in R} c_{ij}(1 + x_{ij}) + \sum_{(v_i, v_j) \in A' \setminus R} c_{ij} x_{ij} \tag{1.14}$$

$$\text{subject to} \quad \sum_{(v_i, v_j) \in R} (1 + x_{ij}) + \sum_{(v_i, v_j) \in A' \setminus R} x_{ij} = \sum_{(v_j, v_i) \in R} (1 + x_{ji}) + \sum_{(v_j, v_i) \in A' \setminus R} x_{ji} \qquad \forall\, v_i \in V' \tag{1.15}$$

$$\sum_{v_i \in S, v_j \in \bar{S}} x_{ij} \geq 1 \quad \forall S = \bigcup_{k \in P} V_k, P \subset \{1,\ldots,p\}, \bar{V} \subseteq S \qquad (1.16)$$

$$\sum_{v_i \in S, v_j \in \bar{S}} x_{ij} \geq 1 \quad \forall S = \bigcup_{k \in P} V_k, P \subset \{1,\ldots,p\}, \bar{V} \subseteq \bar{S} \qquad (1.17)$$

$$x_{ij} \geq 0 \text{ and integer} \qquad \forall (v_i, v_j) \in A' \qquad (1.18)$$

Constraints (1.15) state that the indegree equals the outdegree for all vertices in V'. Constraints (1.16) ensure that the optimal solution connects all the components. Constraints (1.17) are a set of redundant connectivity constraints.

The authors relax constraints (1.15) in a Lagrangean fashion and ignore (1.17) initially, to calculate a lower bound. They solve the subproblem by finding a *Shortest Spanning Arborescence* (SSA) over all the connected components of the graph G'. The lower bound can be strengthened by introducing any violated constraints from (1.17) into the objective function in a Lagrangean fashion. To compute an upper bound, the authors describe the following heuristic. The heuristic first constructs a SSA rooted at any vertex $v_i \in V'$ of the graph G' and connecting the components $G_1,\ldots,G_p$. Let the resulting graph be $\tilde{G} = (V', R \cup \{\text{arcs in SSA}\})$. The graph $\tilde{G}$ is then made symmetric by solving a transportation problem defined as for the directed CPP (See Section 1.2.1.2). Finally the heuristic constructs an Eulerian tour on the resulting graph.

The branch and bound procedure solved 22 out of 23 randomly generated problems with $3 \leq |V| \leq 80$, $24 \leq |A| \leq 180$, $7 \leq |R| \leq 74$, and $2 \leq |p| \leq 8$ to optimality. The heuristic itself seems to perform well. It produced the optimal solution in 10 problems out of 22, and on average, the heuristic solution was within 1.3% of optimality.

1.2.1.6 Stacker Crane Problem

One version of the mixed RPP is called the Stacker Crane Problem (SCP). The SCP is defined on a mixed graph $G = (V, A \cup E)$ where A is the set of directed arcs and E

is the set of edges. The problem is to obtain a least cost circuit which includes each arc in A at least once. The problem can be used to model practical situations like operating a crane or making deliveries. Frederickson, Hecht, and Kim (1978) show how an instance of the TSP can be transformed into an instance of the SCP. Essentially, the SCP is equivalent to the TSP if the cost c_{ij} of every arc is 0. Thus, the SCP is also NP-hard.

Frederickson, Hecht, and Kim (1978) describe two heuristics, LARGEARCS and SMALLARCS, for a mixed graph in which each vertex is incident to at least one arc and the edge costs satisfy the triangle inequality. We provide an outline of the two algorithms below. For a detailed description of the algorithms, see Frederickson et al. If the total cost of the arcs in the graph, C_A, is large relative to the cost of the optimal tour, C^*, the authors use the fact that the cost of an equivalent of a minimum spanning tree for the arcs will be small, to devise the algorithm LARGEARCS. The algorithm basically performs a minimum cost matching on the heads and tails of the arcs. The matching produces a number of disjoint connected components. The algorithm then determines a shortest tree spanning these components using the original edge costs, and finally link the components with two arcs (one in each direction) for each spanning edge. The resulting graph is Eulerian. This algorithm produces a tour whose cost is at most $3C^* - 2C_A$.

The authors propose another heuristic, SMALLARCS, when C_A is small relative to C^*. We already know that when the arc costs c_{ij} are 0, the SCP is equivalent to a TSP. Thus, in this case, it would be ideal to solve the SCP as a TSP. The algorithm shrinks the arcs to nodes and determines a traveling salesman tour on the associated graph. Edges are added between vertices of odd degree. The resulting graph is unicursal and an Eulerian tour can be determined. This algorithm produces a tour whose cost is at most $3C^*/2 + C_A/2$. Since it is difficult to estimate the cost of an optimal tour a priori, the authors suggest applying both algorithms and choosing the tour with the lower cost. In this case, the cost of the tour is at most 9/5 times the cost of the optimal tour.

1.2.2 CAPACITATED ARC ROUTING PROBLEMS

In the capacitated ARPs, the service delivery unit has limited capacity, more than one unit might be required to serve the demand on all arcs or edges. Capacitated ARPs are usually necessary to model real world situations accurately. Golden and Wong (1981) defined the capacitated version of the RPP known as the Capacitated Arc Routing Problem (CARP). In this problem, given an undirected network, associated with each edge (v_i, v_j) is a non-negative demand q_{ij}, in addition to the edge cost c_{ij}. A fleet of m homogenous vehicles with capacity Q is based at the depot. The CARP consists of determining a set of minimum cost cycles servicing all edges with positive demand such that:

(a) every cycle starts and ends at the depot,

(b) the total demand of all edges serviced by any vehicle does not exceed Q, and

(c) each edge with a positive demand is serviced by exactly one vehicle.

If the edge demands q_{ij} are positive for all the edges in the graph, then the problem is referred to as a Capacitated Chinese Postman Problem (CCPP).

Golden and Wong point out that the TSP and the VRP can be considered special cases of the CARP. The constraint that a node must be serviced can be incorporated into the CARP by splitting the node into two nodes joined by an edge with $c_{ij} = 0$ and q_{ij} = demand of the associated node. The authors also show that if the edge costs satisfy the triangle inequality, the 0.5 approximation of the CCPP on a tree network is NP-hard, i.e., the problem of finding a CCPP solution whose cost is less than 1.5 times the optimal solution, is NP-hard. Sahni and Gonzalez (1976) have shown that if the edge costs do not satisfy the triangle inequality, the α-approximate version of the TSP is NP-hard for any finite α. Since the TSP and CCPP are both special cases of the CARP, the results hold for the CARP too.

There are, however specific cases of the CCPP that are well-solved. Assad, Pearn, and Golden (1987) show that the CCPP on a single path with a homogenous fleet and all edge demands equal to 1 is solvable in $O(|V|)$time. They prove a similar

result for the CCPP on a cycle graph with identical demands. They also show that the CCPP on a complete network is solvable if edge demands are less than or equal to $Q/|V|$ if $|V|$ is odd, and $Q/(|V|-1)$ if $|V|$ is even.

Just as the VRP can be transformed into a CARP, Pearn, Assad, and Golden (1987) show that the CARP on an undirected graph can be transformed into a VRP. The authors replace each edge (v_i, v_j) with 3 nodes s_{ij}, m_{ij}, and s_{ji}. The set of nodes in the VRP is $N = \{v_1\} \cup \{s_{ij}, m_{ij}, s_{ji} | (v_i, v_j) \in E \text{ and } q_{ij} > 0\}$, where v_1 is the depot in the CARP. Each of the nodes corresponding to the edge (v_i, v_j) has a demand of $q_{ij}/3$. The distance between the nodes is defined as follows.

$$d(s_{ij}, s_{kl}) = \begin{cases} (c_{ij} + c_{kl})/4 + p(v_i, v_k) & \text{if } (v_i, v_j) \neq (v_k, v_l) \\ 0 & \text{if } (v_i, v_j) = (v_k, v_l) \end{cases}$$

$$d(v_1, s_{ij}) = c_{ij}/4 + p(v_i, v_1)$$

$$d(m_{ij}, v) = \begin{cases} c_{ij}/4 & (v = s_{ij} \text{ or } s_{ji}) \\ \infty & \text{otherwise} \end{cases}$$

where $p(v_i, v_j)$ is the length of the shortest path from v_i to v_j in G. Note that due to the way the distances are defined, any middle node m_{ij} must always be visited by the same vehicle that visits s_{ij} and s_{ji}, and in the sequence s_{ij}, m_{ij}, s_{ji} or s_{ji}, m_{ij}, s_{ij}.

It might not be practical to use this transformation to solve the CARP since the resulting VRP has $3|R|+1$ nodes where $R = \{(v_i, v_j) : q_{ij} > 0\}$ is the number of edges with positive demand. Hence, in the following section, we look at a few exact methods and heuristics for the CARP.

Formulation and Exact Method

Golden and Wong (1981) have proposed the following integer linear programming formulation for the CARP. The decision variables are x_{ijk} and y_{ijk}. x_{ijk} is equal to

1 if edge (v_i, v_j) is traversed from v_i to v_j and 0 otherwise, and y_{ijk} is equal to 1 if edge (v_i, v_j) is serviced while traveling from v_i to v_j and 0 otherwise.

$$\text{minimize} \quad \sum_{k=1}^{m} \sum_{(v_i,v_j)\in A} c_{ij} x_{ijk} \tag{1.19}$$

$$\text{subject to} \quad \sum_{(v_j,v_i)\in A} x_{jik} - \sum_{(v_i,v_j)\in A} x_{ijk} = 0 \qquad \begin{matrix} \forall\ v_i \in V \\ k = 1,\ldots,m \end{matrix} \tag{1.20}$$

$$\sum_{k=1}^{m} (y_{ijk} + y_{jik}) = \begin{cases} 0 & \text{if } q_{ij} = 0 \\ 1 & \text{if } q_{ij} > 0 \end{cases} \qquad \forall (v_i, v_j) \in E \tag{1.21}$$

$$x_{ijk} \geq y_{ijk} \qquad \forall (v_i, v_j) \in E, k = 1,\ldots,m \tag{1.22}$$

$$\sum_{(v_i,v_j)\in A} q_{ij} y_{ijk} \leq W \qquad \forall k = 1,\ldots,m \tag{1.23}$$

$$\left.\begin{aligned} \sum_{v_i,v_j\in S} x_{ijk} &\leq |S| - 1 + n^2 u_k^S \\ \sum_{v_i\in S} \sum_{v_j\notin S} x_{ijk} &\geq 1 - w_p^S \\ u_k^S + w_k^S &\leq 1 \\ u_k^S, w_k^S &\in \{0,1\} \end{aligned}\right\} \qquad \begin{matrix} \forall S \subseteq V \setminus \{v_1\}; S \neq \varnothing; \\ k=1,\ldots,m \end{matrix} \tag{1.24}$$

$$x_{ijk}, y_{ijk} \in \{0,1\} \qquad \forall (v_i, v_j) \in E \tag{1.25}$$

Constraints (1.20) state that at each node, every vehicle that comes in must go out. Constraints (1.21) ensure that only the edges with positive demand are serviced. Constraints (1.22) state that an edge is serviced by a vehicle if and only if it is traversed by that vehicle. (1.23) are the vehicle capacity constraints and (1.24) are the subtour elimination constraints. A subtour in S not connected to the depot could satisfy constraints (1.20)-(1.23), but is not valid. Constraints (1.24) ensure that any cycle traversed by vehicle k in S is connected to $V \setminus S$ and thus to the depot since,

$$\sum_{(v_i,v_j)\in S} x_{ijk} > |S| - 1 \Rightarrow u_k^S = 1 \Rightarrow w_k^S = 0 \Rightarrow \sum_{v_i\in S} \sum_{v_j\notin S} x_{ijk} \geq 1.$$

Belenguer and Benavent (1991) have proposed another formulation using undirected variables. (See Belenguer and Benavent 1991 or Eiselt, Gendreau and Laporte 1995 for a description of the formulation.) They have also derived a number of valid inequalities and have devised a branch and cut procedure to solve the CARP. Using this algorithm, they have solved two relatively small instances ($|V| = 16$, $|A| = 26$ and $|V| = 24$, $|A| = 34$) optimally.

Heuristics

Since the CARP is NP-hard, researchers have developed a wide variety of heuristics and lower bounds for it. The recent survey by Eiselt, Gendreau, and Laporte (1995) provides a good overview of the work done so far in this area. These heuristics can be classified into three categories:

1. *Simple constructive heuristics*: These are one-shot procedures with no local improvement procedures. The five different heuristics that fall under this category are as follows:
    - The construct-strike algorithm proposed by Christofides (1973) and, modified by Pearn (1989). This algorithm successively constructs feasible cycles and removes them from the graph until all edges are covered. The Christofides version uses a 1-Matching algorithm to keep the graph connected at all times, while the Pearn version uses a minimum spanning tree algorithm to connect the several components of the graph, and then the matching algorithm to generate an Euler cycle.
    - The path-scanning algorithm developed by Golden, DeArmon, and Baker (1983), and modified by Pearn (1989). The Golden et al. algorithm constructs feasible cycles one at a time using one of five different myopic optimality criteria. The solution to the problem is the best among the five solutions. Pearn uses a criterion at random, at each step of cycle construction.

- The augment-merge algorithm by Golden, DeArmon, and Baker (1983) which initially constructs a different cycle for each edge to be serviced, and then combines different cycles using a savings criterion.
- The parallel-insert algorithm by Chapleau et al. (1984). This algorithm is part of a school bus routing system which aims at reducing zigzag routes in addition to minimizing the total routing cost. In order to achieve this objective the algorithm that works like the path scanning algorithm, constructs several routes in parallel.
- The augment-insert algorithm by Pearn (1991). Pearn developed this algorithm specifically for sparse graphs with large edge demands. The algorithm first creates feasible cycles using either a cost or a demand criterion as long as it is possible. Once feasible cycles cannot be found, the remaining edges are inserted into the existing cycles using a savings criterion.

2. *Two-phase constructive heuristics*: The two types of heuristics in this category develop cycles in two phases. The cluster-first, route-second heuristics (Win 1987, Benavent et al. 1990) first group the edges into clusters, each having a weight of at most Q, and construct a vehicle route for each cluster during the second phase. On the other hand, the route-first, cluster-second heuristics (Win 1987, Ulusoy 1985) construct a giant Euler tour over all edges with positive demand first, and then partition the tour into feasible clusters.
3. *Improvement heuristics*: These are post-optimization heuristics that can be applied to the solution obtained using any of the heuristics mentioned above. Ulusoy (1985) mentions one particular scheme in his application. Hertz, Laporte and Nanchen (1996) have developed a number of improvement procedures for the undirected RPP which can be directly applied to the individual routes of a CARP solution. Hertz, Laporte and Mittaz (1996) describe a procedure called POSTOPT that combines all the routes of a CARP solution into a single tour and then cuts this giant tour into smaller routes satisfying the capacity constraints, while trying to determine a new CARP solution with a lower objective value.

4. *Metaheuristics*: Hertz, Laporte and Mittaz (1996) have developed CARPET, a tabu search based heuristics for the CARP. CARPET minimizes the total cost of traversing all the cycles. It incorporates some of the concepts contained in TABUROUTE (Gendreau, Hertz, and Laporte 1994), a tabu search algorithm for the VRP, and the improvement heuristics described in Hertz et al. (1994) and Hertz et al. (1996). The basic move consists of moving an edge that requires service from it current route to another route. The heuristic contains several features of tabu search such as random tabu tags, self-adjusting penalties, diversification by frequency counts, and intensification by applying the POSTOPT local improvement procedure if the best known solution is not updated for a fixed number of iterations. Computational results indicate that the algorithm is robust and produces the optimal solution for several benchmark problems.

The performance of the heuristics can be measured in terms of the heuristic solutions' deviation from the optimal solutions. Since it is difficult to obtain optimal solutions for most instances of the CARP, most researchers use lower bounds to get an estimate on the optimality gap. The following section describes several lower bounds for the CARP.

Lower Bounds

Benavent, Campos, Corberán, and Mota (1992) provide a review and comparison of all the lower bounds that researchers have developed for the CARP. Golden and Wong (1981) were the first to suggest a matching based lower bound (*MLB*). In this bound, the authors use $\underline{m} = \left\lceil \sum_{(v_i,v_j)\in E} q_{ij} \Big/ Q \right\rceil$ as a lower bound on the number of vehicles needed in any solution to the CARP. Since any solution to the CARP has at least $\underline{m}$ cycles, each passing through the depot, the degree of the depot has to be increased from d_1 (degree of the depot) to $2\underline{m}$. Hence, the basic idea is to construct a matching graph and obtain a minimum cost augmentation of G to ensure that the graph is even.

Let $\lambda = 2\underline{m} - d_1$ if d_1 is even, and $2\underline{m} - d_1 - 1$ if d_1 is odd. The matching graph consists of S, the set of vertices in G that have an odd number of edges with non-zero demand incident to them, and two sets of artificial vertices $A = \{a_1,\ldots,a_\lambda\}$ and $B = \{b_1,\ldots,b_\lambda\}$. Each of the artificial vertex represents a copy of the depot. The authors define a matching graph $H = (S \cup A \cup B, D)$ and define the cost of the edges c_{ij} as follows:

- If $v_i, v_j \in S$, then c_{ij} = length of the shortest path between nodes v_i and v_j.
- If $v_i \in S$ and $v_j \in A$, then c_{ij} = length of the shortest path between the depot and v_i.
- If $v_i \in A$ and $v_j \in B$, then c_{ij} = the length of the shortest edge out of the depot.
- If $v_i \in B$ and $v_j \in B$, then $d_{ij} = 0$.

The basic idea is to match vertices of odd degree to nodes in set A (copies of the depot). If all the nodes in A are not paired with nodes in S (since some of the nodes in S might be matched with one another), then they are matched with nodes in B. Finally, any unmatched nodes in B are matched with each other. We do not connect the nodes in A with one another using edges of length 0, since this would be counter-productive to the idea of matching nodes in S to the depot. If $z(H)$ is the value of the minimum cost 1-matching on the matching graph, then Golden and Wong show that

$$MLB = z(H) + \sum_{\substack{(v_i,v_j)\in E \\ q_{ij}>0}} c_{ij} \qquad (1.26)$$

is a valid lower bound for any optimal solution to the CARP.

Assad, Pearn, and Golden (1987) describe a node scanning lower bound (*NLB*) for the CCPP. In this case, the authors increase the degree of the depot by adding minimum cost paths from the nodes to the depot. They add the shortest path first and continue in the increasing order of path lengths. Let c_{ij} denote the length of

the shortest chain between nodes v_i and v_j and d_i denote the degree of node v_i, and let vertex v_1 denote the depot. The algorithm can be described as follows.

- Renumber all the vertices in non-decreasing order d_{1i}.
- Let $l = \min\left\{k : \sum_{i=2}^{k} d_i \geq \lambda\right\}$ where λ is defined as in the matching based lower bound. Reset $d_l = \lambda - \sum_{i=2}^{l-1} d_i$ and calculate *NLB* as

$$NLB = \sum_{(v_i, v_j) \in E} c_{ij} + \sum_{i=2}^{l} d_{1i} d_i \tag{1.27}$$

Since we need to add λ paths to the given graph, we start with the vertex v_i with the lowest value of d_{1i} and add d_i paths from this vertex to the depot. We continue the process until λ paths are added. The result can be extended to the CARP by considering the graph induced by the edges with positive demand. Pearn (1988) has proposed a lower bound that combines the ideas behind these two methods and shows that it dominates *MLB*. Win (1987) has also developed several lower bounds that dominate *MLB*.

Benavent et al. (1992) have compared all these bounds and have developed four different lower bounds. Three of these bounds require solving a minimum cost matching problem, and the fourth is a dynamic programming algorithm producing a lower bound. Their computational results show that one of their lower bounds that solves a minimum cost matching problem, *LB2* outperforms the others based on quality of the bound and computational time. *LB2* and the bounds proposed by Win improve upon *MLB* by considering the number of vehicles required for certain subgraphs, and not just the vehicles necessary to cover the whole graph. For a detailed discussion of all the bounds, see Benavent et al. In the next section, we provide an overview of several application areas for ARPs. For each application area, we present a few of the significant real-world applications.

1.2.3 APPLICATIONS OF ARC ROUTING

Arc routing problems have a wide variety of applications. Most problems occur in the delivery of services like snow plowing, garbage collection, and mail delivery. In most applications, it is not necessary to service all arcs of the graph. In addition, more than one vehicle or person is necessary to deliver the service and several additional constraints might have to be incorporated. Considering all these, the most useful problem for modeling real world situations is the CARP. However, the RPP and the CPP might be subproblems in many situations and an understanding of these problems would help in developing algorithms.

The CPP has been used to model practical problems in a few situations. Malek, Mourad, and Pandya (1989) show how the topological testing of computer systems can be modeled as a CPP. The authors form a graph in which the nodes represent the hardware (processors, registers, etc.) for the specific level at which testing is to be done, and the arcs represent the data flow. They achieve the objective of sending a testing packet traversing all the arcs and vertices in minimal time by finding an Eulerian circuit on the underlying graph.

Barahona (1990) describes several applications that are similar to the CPP. One of the problems is in the design of VLSI circuits. Here, the chip components are arranged in two layers. Wires connecting the components must be assigned to the layers such that they do not cross each other on the same layer, but may go from one layer to another through special connections called vias. The objective is to minimize the number of vias. The author shows that one can construct a planar graph in which the vertices correspond to wire end points and crossings and edges correspond to sections of wire between these. The number of vias is minimized by determining the smallest number of edges to remove to make the graph bipartite.

Garbage Collection

Garbage collection is one of the most essential and common applications of arc routing. Beltrami and Bodin (1974) discuss the overall procedure adopted for

developing garbage collection routes for New York City. They first address the issue of location of dumpsites and assignment of days to streets. Then they form the actual collection routes to minimize total deadheading. They use a route-first, cluster-second heuristic. Bodin, Fagin, Welebny, and Greenberg (1989) give the computer implementation details of this project. Clark and Gillean (1975) and Clark and Lee (1976) describe the results of a similar study conducted in Cleveland, Ohio. McBride (1975) addresses the problem of reducing the number of left and U-turns while routing garbage collection trucks. Turner and Hougland (1975) present another study performed at Blacksburg, Virginia.

Mail Delivery

ARPs in the context of mail delivery have been studied well in Canada and the United States. Roy and Rousseau (1989) explain that in the Canadian Postman Problem, the letter carrier starts and ends his work day at the post office. From the post office, each postal carrier travels to a depot to start his route. Travel times to and from the post office are included in the work day. The routes are not constrained by the capacity of a postal bag since relay boxes are conveniently located along the route. Bouliane and Laporte (1992) study the problem of locating relay boxes. In addition to the routing, since the problem deals with the location of the depots for each carrier, it is viewed as an arc oriented location-routing problem. Levy and Bodin (1988, 1989) also view the US post problem as a location-routing problem. In this case, the concept of relay boxes does not exist. So, routes have to be developed taking into consideration the maximum load of a postal bag. Levy and Bodin describe that the problem consists of partitioning the arcs into balanced clusters and then locate depots. The letter carrier drives to a depot, delivers mail to an adjacent cluster, returns to the depot, and continues with all adjacent clusters in a similar fashion. He then drives to another depot and continues till the end of the work day.

Meter Reading

In this application, meter readers periodically visit and read the meters of customers in their service area. Stern and Dror (1979) describe the problem as an *m*-postman problem where the objective is to minimize *m* and design routes that satisfy maximum duration constraint. Their heuristic algorithm provides a 40% reduction in the number of tours on data from the city of Beersheva in Israel. Wunderlich, Collette, Levy and Bodin (1992) describe a computerized system implemented for the Southern California Gas Company.

School Bus Routing

Another application area is school bus routing. Here the objective is to minimize the number of buses and the total distance traveled by all the buses. This problem usually involves a number of additional constraints on several issues such as the number of students in the bus, the time spent by a student in the bus, and student mix, just to mention a few. Desrosiers, Ferland, Rousseau, Lapalme, and Chapleau (1986) have developed an algorithm based on column generation to schedule buses for 60 schools and 20,000 students. Braca, Bramel, Posner and Simchi-Levi (1993) describe a computerized system for routing and scheduling school buses throughout the five boroughs of New York city. Earlier studies on school bus routing include Bennet and Gazis (1972), Bodin and Berman (1979), and Swersey and Ballard (1984).

Snow Plowing

Snow plowing is an interesting and important arc routing application. Typically, the roads have different priority levels and the roads with higher priority have to be cleared before the roads with lower priority. This problem has been defined as the Hierarchical Postman Problem by Dror, Stern, and Trudeau (1987). Haslam and Wright (1991) have developed an algorithm for snow and ice control in Indiana. Lemieux and Campagna (1984) describe a similar problem and address several additional issues such as the composition and size of the fleet and the number and

location of service centers. Cook and Alprin (1976) have developed an algorithm for routing salt spreader trucks in Tulsa, and Eglese and Li (1992) have studied the gritting operations for the Lancashire County Council in England.

Street Sweeping

One of the unique aspects of street sweeping is that streets can be swept only during particular time slots when parking is prohibited on the street. Bodin and Kursh (1978, 1979) describe a study performed in New York City and Washington, D. C. They have developed a computerized system to develop routes for a given time slot such that the work is balanced among the fleet of sweepers and all streets are provided sufficient coverage. Eglese and Murdock (1991) present a study where the availability of the streets for sweeping is not constrained by parking regulations. They point out that in their study all streets can be considered as two way streets as opposed to the study by Bodin and Kursh.

1.2.4 STOCHASTIC NODE ROUTING

Most of the current research on arc routing addresses ARPs in a deterministic context. In this thesis, we define the Stochastic Eulerian Tour Problem (SETP) and investigate several characteristics of the problem and develop heuristics to obtain good solutions. In order to gain an understanding of the work done on the equivalent node routing problem, we present a brief summary of the literature on the Probabilistic Traveling Salesman Problem (PTSP). For a recent survey of most stochastic VRPs, see Gendreau, Laporte and Séguin (1996).

The stochastic version of the TSP arises when some elements of the problem are random. For example, the travel times between nodes can be stochastic or the set of customers or nodes to be visited can be random. We concentrate on the TSP with stochastic customers. Generally, stochastic programs are modeled in two stages. The first stage consists of determining an a priori solution to the problem. For any given instance of the problem, in the second stage, a corrective action or recourse is applied

to tailor the first stage solution. The cost of this action is also generally figured while determining the a priori solution that minimizes the total expected cost. This is the basic idea behind a stochastic program with recourse.

Jaillet (1985) introduced the TSP with stochastic customers as the PTSP. It is essentially a TSP where each vertex v_i is present with a probability p_i, and hence the number of vertices to be visited is a random variable. Consider a problem of routing through a set of n known points. On any given instance of the problem, one needs to visit only a subset consisting of k $(0 \leq k \leq n)$ points. The recourse action Jaillet uses is to follow an a priori tour and simply skip absent customers. Under the assumption that $p_i = p$ for all vertices, Jaillet derives closed form expressions for computing the expected length of a tour. He also derives bounds and several interesting properties of the problem. Most of the results in Jaillet (1985) are summarized in Jaillet (1988), and Jaillet and Odoni (1988).

Jaillet shows that an optimal TSP tour can be arbitrarily bad for the TSP with stochastic customers. He also shows that an optimal tour for the TSP with stochastic customers may intersect itself in the Euclidean plane. This is in contrast to what holds for optimal TSP tours. These results indicate that algorithms have to be developed specifically with the stochastic problem in mind. Jaillet has developed a number of heuristics by suitably modifying several well-known TSP heuristics such as the Clarke-Wright (1964) algorithm and tour merging algorithms. Rossi and Gavioli (1986) present computational results after having tested three of Jaillet's heuristics.

Bertsimas (1988) and Bertsimas and Howell (1993) have developed a few more heuristics based on probabilistic 2-opt edge exchange, vertex moves within a tour, and space filling curves (Bartholdi and Platzman 1982). Laporte and Louveaux (1993) have developed a branch and cut algorithm called Integer L-Shaped method that is applicable to many stochastic programs with recourse. Laporte, Louveaux and Mercure (1994) have applied this method to the stochastic TSP and solved instances with up to 50 vertices optimally.

1.3 OBJECTIVES OF THIS RESEARCH

The general objective of ARPs is to determine a least cost traversal of a specific subset of edges of the graph, with or without additional constraints. As described earlier, these problems occur in a wide variety of practical contexts. This research addresses two specific problems in arc routing that have excellent application potential.

The first problem that we consider is the CARP. The CARP is one of the most important problems in arc routing due to its presence in applications such as snow plowing, street cleaning, garbage collection, mail delivery, and many others. The CARP is a very hard problem, and it is quite unrealistic to believe that exact procedures can be used to solve even average sized problems. Researchers have developed several heuristics for the CARP. Most of these heuristics are simple one-shot heuristics. Also, each heuristic tends to perform well on specific types of graphs. As described earlier, Hertz, Laporte and Mittaz (1996) have developed a tabu search based heuristic, CARPET, for the CARP. This incorporates several local improvement procedures and outperforms the earlier heuristics for the CARP.

Tabu search first proposed by Glover in 1986, is a metaheuristic that makes use of memory structures and exploration strategies based on information stored in memory to search beyond local optima. (See Glover 1989, 1990 and Glover, Taillard, and de Werra 1993 for recent overviews.) Here, the procedure repeatedly moves from one solution to the best among its neighboring solutions.

The objective of CARP is to produce a solution that has the minimum traversal cost. However, in many practical situations, in addition to minimizing the total cost, a secondary objective of balancing the total work load (demand) fairly equally among the routes plays an important role. We consider this secondary objective also in our heuristic and try to produce solutions that balance the work load at the least cost.

We feel that it is quite important to incorporate this feature, since most applications such as mail delivery, meter reading, and garbage collection require work load balancing. In a real world scenario, if the existing algorithm considers only the

total cost, generally, the planner revisits the solution and moves demand among the various routes myopically in order to balance the load. On the other hand, our procedure aims to be a little more global and builds this additional feature into the algorithm, and hence can serve as a useful planning tool for several arc routing applications.

The second problem that we consider in this thesis is the *Stochastic Eulerian Tour Problem* (SETP). The SETP arises when the set of edges that have to be visited on any particular day is random. The investigation of this problem was actually motivated by the existence of a real world problem. In the UK postal system, the carriers deliver mail a second time in the afternoon when the number of streets to be visited is very small and varies from day to day. Given this scenario, the mail carrier, while following his regular route, usually skips the streets that do not require a visit.

Given an Eulerian graph, it is important to note that there may be more than one Eulerian tour for a given graph. However, all these tours have the same cost and hence there is no optimization involved in the deterministic Eulerian Tour Problem. However, for the SETP, each tour has certain advantages and disadvantages with respect to skipping edges, and thus has different expected lengths. Thus, given an undirected graph $G=(V,E)$ where all the edges in $E\left(|E|=n\right)$ require service, a distance matrix D, and a probability distribution for the number of required edges present, the SETP seeks an a priori Eulerian tour of minimum expected length.

The SETP has not been investigated in the literature thus far. We feel that it plays an important role in scenarios where the number of edges to be visited each day is random and smaller compared to the total number of edges that require service. This motivates our investigation of this problem and its properties. We show that the SETP is NP-hard, and hence it is not possible to solve realistic sized problems optimally using algorithms that would run in polynomial time. Hence, this thesis also concentrates on the development of heuristic algorithms specifically for the SETP.

1.4 ORGANIZATION OF THE THESIS

As an introduction and motivation for the thesis, this chapter provided a review of some of the basic literature in arc routing, and additional literature that is most relevant to the proposed research, and finally the objectives of the three papers. The review began with the origins of research on ARPs and then presented an overview of the literature on uncapacitated ARPs. While the undirected and directed CPP are solvable in polynomial time, the other uncapacitated ARPs are combinatorially hard. Exact methods and heuristics have been developed for these problems using the conditions for unicursality. Most of the exact methods are adaptations of algorithms for NRPs, and do not seem to perform as well as they perform on the NRPs.

The next section presented a discussion of the CARP and related research. It highlighted the CARP as an important and difficult problem and described several lower bounds and heuristics. The following two sections discussed the literature on arc routing applications and the PTSP.

This literature review attempted to provide an understanding of the research done in the area of arc routing. The overview of arc routing applications helps us understand the prevalence of CARP applications in several everyday problems. Many of the applications also have the inherent requirement of developing fairly equally loaded cycles or route. This motivated the first problem that we address in this thesis.

Chapter 2 presents TABUCARP, the tabu search based heuristic for the CARP, that considers work load balancing as a secondary objective. TABUCARP drops (adds) one or more edges from a route that is over (under) capacity to a neighboring route. The algorithm continuously moves capacity excesses and deficits towards the depot. We describe the several features such as self-adjusting penalties, random tabu tags, and adaptive memory incorporated in our algorithm. We also present computational results on a set of benchmark problems and another set of random problems. The results indicate that while our solutions are similar to CARPET's solutions on the benchmark problems, the total distance traversed by TABUCARP solutions for the random problems is 2.78% greater when compared to that of CARPET solutions. However, this deterioration in the objective function

value is marginal when we consider the fact that the routes produced by TABUCARP are better balanced than the routes produced by CARPET.

We define the SETP in Chapter 3. We derive a closed form expression for the expected length of a given tour when the number of present edges follows a binomial distribution. We also show that the SETP is NP-hard, even though the deterministic counterpart is solvable in polynomial time. We derive further properties and a worst case ratio for the deviation of the expected length of a random Eulerian tour from the optimal tour in the expected sense. Finally, we present some of the desirable properties in a good a priori tour using illustrative examples.

Chapter 4 describes three heuristics for the SETP. The first heuristic is a simple greedy heuristic that determines the next edge to service based on the increase in the expected length by adding that edge. Finally, based on the order of visiting the edges, the heuristic constructs the actual Eulerian tour. The second heuristic is a modification of the first heuristic. This heuristic constructs the actual tour while adding the edges. It starts at the designated depot and determines the next edge to service among the edges adjacent to the depot, as the one that results in the minimum increase in the expected length when appended at the end of the tour. The process continues until all edges are added to the tour. The third heuristic takes advantage of the results from Chapter 3. The heuristic constructs several small sub-tours and then concatenates the sub-tours to form the a priori Eulerian tour. We also use a post-optimization procedure that is a modification of the US procedure proposed by Gendreau et al. (1992) for the TSP.

We present computational results comparing the performance of the three heuristics and also, comparing the expected length of the tours with that of a random Eulerian tour. Our computational results indicate that the sub-tour construction heuristic along with the post-optimization procedure consistently produces good tours for grid networks, while the second heuristic and the post-optimization procedure seems to be better for general Euclidean networks. Finally, Chapter 5 summarizes the results and contributions of this thesis and presents directions for future research.

CHAPTER 2

TABUCARP: A TABU SEARCH ALGORITHM FOR THE CAPACITATED ARC ROUTING PROBLEM WITH WORK LOAD BALANCING

2.1 INTRODUCTION

Arc routing problems play an important role in distribution management, and occur in a wide variety of practical problems such as mail delivery, school bus routing, snow clearance, street sweeping, garbage collection, and several others where streets have to be traversed for performing work. This chapter presents a tabu search heuristic for the Capacitated Arc Routing Problem (CARP), one of the most important problems in the area of arc routing. We can define the CARP formally as follows. Let $G = (V, E)$ be a graph where $V = \{v_0, v_1, \ldots, v_n\}$ is the vertex set and $E = \{(v_i, v_j) : i \neq j\}$ is the edge set. A fleet of m identical vehicles of capacity Q is based at the depot v_0. The value of m is either fixed at a constant or bounded above by $\overline{m}$. Each edge (v_i, v_j) has a non-negative demand q_{ij} and cost c_{ij} associated with it. The CARP consists of determining a set of minimum cost cycles traversing all edges with positive demand such that:

(a) each cycle starts and ends at the depot,

(b) the total demand of all edges serviced by any vehicle does not exceed Q, and

(c) each edge with a positive demand is serviced by exactly one vehicle.

Note that a vehicle may traverse an edge without servicing it.

As mentioned above, the CARP (usually with additional side constraints) occurs in many practical day-to-day problems. In several realistic situations, in addition to developing minimum cost solutions, the concept of balancing the work load among the cycles plays an important role. For example, when developing routes for letter carriers or meter readers, the planner would like to assign a fairly equal amount of work to each member or the crew. The union regulations might also require this. Hence, it becomes an important practical consideration. Hence in our research, we consider work load balancing as a secondary objective for the traditional CARP.

The CARP is known to be NP-hard, and hence, researchers have developed a wide variety of heuristics and lower bounds for it. The recent survey by Eiselt, Gendreau, and Laporte (1995) provides a good overview of the work done so far in this area. We also provided a brief review of the several simple and two phase constructive heuristics and the lower bounds for the CARP in the previous chapter. The chapter also provided a description of CARPET, a tabu search based heuristic for the CARP developed by Hertz, Laporte, and Mittaz (1996). CARPET minimizes the total cost of traversing all the cycles. Computational results indicate that the algorithm is robust and produces the optimal solution for several benchmark problems. However, their basic moves that guide the search do not consider the additional factor of balancing the workload among the various cycles.

Tabu Search (TS) is a metaheuristic that makes use of memory structures and exploration strategies based on information stored in memory to search beyond local optima. (See Glover 1989, 1990, Glover and Laguna 1997 and Glover, Taillard, and de Werra 1993 for recent overviews.) Here, the procedure repeatedly moves from one solution to the best among its neighboring solutions. The procedure accepts non-improving moves at certain circumstances to get away from local optima. In this situation, to prevent cycling, certain moves are temporarily forbidden and inserted into short term memory in a tabu list which is updated constantly. Two important strategies that use long term memory to improve the search process are intensification and diversification. Intensification is based on the idea that a good solution will be

more likely to lie within a promising region. An example of simple intensification can be simply returning to the best solution found so far and search around it more thoroughly. The purpose of diversification is to cover larger regions of the solution space. Diversification can be carried out simply by partially or fully re-starting the search process. Frequency-based memory can be useful in diversifying the search by penalizing frequently performed moves so that exploration towards regions not visited in previous search will be encouraged.

Researchers have used TS successfully to find very good solutions to several combinatorially difficult problems. Three main areas of application are production scheduling, graph theory, and vehicle routing. In the area of production scheduling, a number of TS heuristics have been developed for the flowshop sequencing problem (Widmer and Hertz 1989, Taillard 1990, Daniels and Mazola 1993). Dell'Amico and Trubian (1993) and Brandimarte (1993) have developed tabu heuristics for the job shop scheduling problem. Hertz and de Werra (1987) used TS techniques for graph coloring. Friden, Hertz, and de Werra (1989, 1990) have developed a very good TS heuristic for finding stable sets in large graphs and a TS based exact algorithm for determining the maximum independent set in a graph. Gendreau, Soriano, and Salvail (1993) present an efficient TS heuristic for finding maximum cliques in a graph. In the area of vehicle routing, several TS heuristics have been developed for the vehicle routing problem (VRP) and the VRP with time windows. Several researchers have developed TS heuristics for the VRP including Pureza and França (1991), Osman (1991, 1993), Semet and Taillard (1993), Taillard (1992), Gendreau, Hertz, and Laporte (1994), and Rego and Roucairol (1996). Some of these algorithms have produced impressive results. For the VRP with time windows, the TS heuristics by Potvin, Kervahut, Garcia, and Rousseau (1996) and Taillard, Badeau, Gendreau, Geurtin, and Potvin (1997) produce near optimal solutions. As mentioned earlier, Hertz, Laporte, and Mittaz (1996) have developed an efficient tabu search heuristic for the CARP.

In this chapter, we present TABUCARP, a tabu search heuristic for the CARP. There are several differences between TABUCARP and CARPET. TABUCARP

considers the additional objective of balancing work load among the various cycles. Hence the basic move strategies are different for the two algorithms. We present a detailed description of the basic definitions and the various components of the neighborhood search technique in Section 2.2. Section 2.3 presents a description of the TABUCARP heuristic. We provide the computational results in Section 2.4, and conclusions in Section 2.5.

2.2 TABU SEARCH TECHNIQUE

The central idea of any TS heuristic is the iterative local search procedure that moves from a current solution to one of its neighbors. This local search advances keeping the objective of the problem in perspective. The overall objective of our problem is to develop minimum cost feasible cycles such that the work load (demand) is balanced among the cycles. In order to achieve this objective, we try to move the capacity excess or deficit from one cycle to another neighboring cycle. We start with the cycles that are farthest from the depot and work towards the depot. Thus, we try to move all the deficits and excesses among the cycles while attempting to balance the work load among cycles.

With this overall methodology in mind, we define the following to help us describe the basic procedures that we use in our search. A solution S to the CARP consists of m cycles $C_1, C_2, \ldots, C_m$. A solution might contain empty cycles also. If all the m cycles of a solution S satisfy the capacity constraint, then solution S is feasible. If one or more cycles violate the capacity constraint, the solution S is infeasible. Edges that are serviced by cycle C_k are called service edges on cycle C_k, and all the other edges on the route that are just traversed are called non-service edges on cycle C_k. Cycle C_k is represented as $C_k = \left(P_{out(k)}, P_k, P_{in(k)}\right)$, where

$P_{out(k)}$ = the path from the depot to the first service edge on cycle C_k, containing only non-service edges

P_k = the path from the first to the last service edge on cycle C_k

$P_{in(k)}$ = the path from the last service edge on cycle C_k to the depot, containing only non-service edges.

Let $d_{in(k)}$ and $d_{out(k)}$ be the length of the paths $P_{in(k)}$ and $P_{out(k)}$ respectively. Note that an edge may appear more than once in a cycle. Let n_{ijk} be the number of times an edge (v_i, v_j) appears in a cycle C_k, and E_k be the set of edges with positive demand that are serviced on cycle C_k. For each cycle, we also define a distance measure, d_k, to keep track of the cycle's proximity to the depot. We define the distance measure of cycle C_k as $d_k = \max(1, \max(d_{in(k)}, d_{out(k)}))$. We next describe some of the main components of the tabu search procedure.

2.2.1 OBJECTIVE FUNCTION

The objective function value $F_1(S)$, of any feasible solution S is

$$F_1(S) = \sum_k \sum_{(v_i,v_j)\in C_k} n_{ijk}\, c_{ij} \qquad (2.1)$$

where $(v_i, v_j) \in C_k$ means edge (v_i, v_j) is part of cycle C_k. We define a cycle C_k to be violating capacity if the total demand served on this cycle is more than the capacity Q or less than a threshold t, where t is a parameter. The tabu search procedure allows capacity violation at a cost. In this case, we define the cost of violation as the weighted sum of the distance measure times a penalty for capacity violation, over all cycles. For a solution violating capacity, let F be the set of cycles that are over capacity and D be the set of cycles with demand less than t. In case of a solution S violating capacity, the objective function value $F_2(S)$ can be written as

$$F_2(S) = F_1(S) + \sum_{C_k \in F} d_k \alpha_k \left(\sum_{(v_i,v_j)\in E_k} q_{ij} - Q \right) + \sum_{C_k \in D} d_k \beta_k \left(t - \sum_{(v_i,v_j)\in E_k} q_{ij} \right) \qquad (2.2)$$

where α_k and β_k are positive penalties for cycles violating capacity. α_k denotes the penalty for the cycles that are over capacity and β_k denotes the penalty for cycles serving demand less than t. Note that when a solution S is feasible and the cycles are balanced, $F_1(S) = F_2(S)$. Let F_1^* and F_2^* denote respectively the lowest values of $F_1(S)$ and $F_2(S)$ obtained during the search, S^* be the best known feasible solution, and $\tilde{S}^*$ be the best known solution (feasible or not).

2.2.2 PENALTY FUNCTION

The penalty function is of the form shown in Figure 6. The function consists of two parts – one corresponding to cycles that are over capacity and the other to cycles that serve a total demand of less than t. The penalty for over capacity is more than the penalty for under capacity.

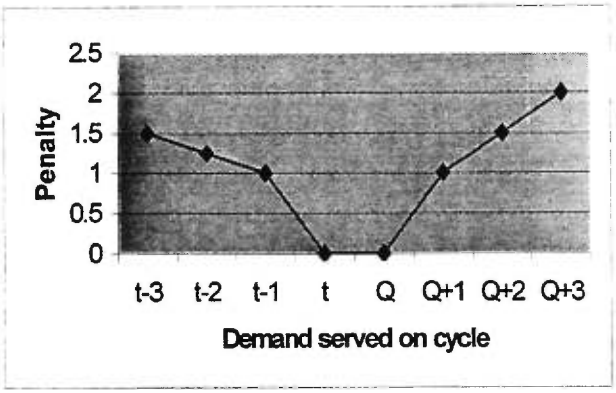

Figure 6. Penalty Function

Note that the slope of the penalty function for cycles that are over capacity (0.5) is twice that of slope of the penalty function for cycles that are under capacity (0.25). If the demand served by a cycle C_k is in the interval $[t,Q]$, that cycle does not incur any penalty. If the demand served is $(t-1)$ or $(Q+1)$, the cycle incurs a penalty of 1. If the demand served is less than $(t-1)$ or greater than $(Q+1)$, then the penalty

incurred depends on the demand served on the cycle and the slope of the function. We set the threshold $t = \lceil 0.9\,Q \rceil$.

We also use self-adjusting penalties as in Gendreau et al. (1994). Every λ iterations, if all previous λ iterations were feasible, we set $\alpha = \alpha / 2$ and $\beta = \beta / 2$, and thus the penalty coefficients are halved. We double the penalty coefficients by setting $\alpha = 2\alpha$ and $\beta = 2\beta$ if all of the previous λ iterations were infeasible. This helps the search procedure to produce a mix of feasible and infeasible solutions. Based on initial experimentation, we set the value of $\lambda = 5$ in our implementation.

2.2.3 BASIC MOVES

The basic move in our heuristic consists of moving a path with at least one service edge from the current cycle to one of the other cycles in the solution. In the rest of this chapter, we refer to a cycle that is over capacity as an excess cycle and a cycle that is under capacity as a deficit cycle. Given a solution *S*, we move to a neighboring solution using one of the following moves:

- *drop move*: move a path containing at least one service edge out of an excess cycle into one of its neighboring cycles;
- *add move*: move a path containing at least one service edge into a deficit cycle from one of its neighboring cycles.

Next we describe the steps involved in the drop and add moves in detail.

Given a solution *S*, we pick the cycle with the maximum distance measure as the candidate cycle C_k and a drop or add move is performed based on the total demand served by this cycle. We perform a drop move if the candidate cycle is an excess cycle and an add move if it is a deficit cycle. If the cycle with the maximum distance measure is at capacity, we pick the next farthest. If all the cycles are at capacity, we perform a *perturb move* so that the search will not be stalled. The perturb move picks the cycle with the least distance measure as the candidate cycle and drops the service edge with the least demand from this cycle into one of its neighboring cycles, and the search continues.

It is important to define the neighborhood of a solution precisely in order to guide the search process. At each step, one of the options is to perform an exhaustive search of the neighborhood. But this option could be very expensive in terms of computation times. So, we evaluate a reduced neighborhood of the solution at each step. However, we have to be careful not to limit the size of the neighborhood excessively, since this could eliminate good solutions from being examined. We next describe the steps used to define the structure of the neighborhood in TABUCARP.

2.2.4 NEIGHBORHOOD STRUCTURE

Given a solution S and a candidate cycle C_k in S, we consider only a subset of all the cycles in S for evaluating the next move. Specifically, we choose the set of all cycles C_l such that $|d_k - d_l| \leq \delta$, where δ is a parameter, as the set of neighboring cycles of cycle C_k. If the given solution S has less than m cycles, we include an empty cycle in the set of neighboring cycles. We define parameter δ as the average distance of all service edges from the depot, where the distance of an edge from the depot is the maximum of the shortest path distances to its extremities.

If the candidate cycle C_k is an excess cycle, then we have to perform a drop move. We consider all the paths starting with an edge served on the candidate cycle and with total demand on the path less than or equal to w, where w = (total demand served on the candidate cycle C_k – the threshold t). Among these paths, we randomly choose q paths for evaluation. For each one of the q paths, we drop the path from cycle C_k and add it to each one of the neighboring cycles in turn using the DROP and ADD procedures described in Hertz et al. (1996b). We provide a summary of the DROP and ADD procedures at the end of this sub-section. The set of solutions thus generated forms the neighborhood of the current solution S.

If the candidate cycle C_k is a deficit cycle, then we have to perform an add move. In this case, we randomly choose q paths from each of the neighboring cycles for evaluation. The candidate paths have to start with an edge serviced on the

neighboring cycle and the total demand on the path has to be less than or equal to w, where $w = (Q$ – total demand served on cycle $C_k)$. We drop each of the q paths from the neighboring cycles and add it to the candidate cycle C_k, and the neighborhood of the current solution S consists of the set of all solutions generated thus.

The DROP and ADD procedures use another procedure termed SHORTEN. Given a cycle $C_k = (v_0 = v_{i_1}, v_{i_2}, \ldots, v_{i_t} = v_0)$ starting and ending at the depot and serving a set of edges R_k, this procedure tries to identify a shorter cycle C_k' starting and ending at the depot and serving the same set of edges, but not necessarily in the same order. We introduce an artificial depot v_0' connected to v_0, and a service edge (v_0, v_0') of zero demand and zero cost, so that v_0 does not get deleted while performing the SHORTEN procedure. We first provide a detailed description of procedure SHORTEN.

Procedure SHORTEN

Step 1: Set $r \leftarrow 1$ and $C_k' \leftarrow C_k$.

Step 2: Set $s \leftarrow 1$ and $C_k' = (v_{j_1} = v_{i_r}, \ldots, v_{j_{t-r+1}} = v_{i_t}, \ldots, v_{j_t} = v_{i_r})$. Let $b_{ij} = 0$ for all edges $(v_i, v_j) \in C_k'$, and a_{ij} = the number of times an edge $(v_i, v_j) \in R_k$ appears in C_k'.

Step 3: If $(v_{j_s}, v_{j_{s+1}}) \notin R_k$, go to Step 4. Else, if $b_{j_s, j_{s+1}} < a_{j_s, j_{s+1}} - 1$, then increment $b_{j_s, j_{s+1}}$ and b_{j_{s+1}, j_s} by 1 and go to Step 4. Otherwise, attempt to find the smallest index $h > s$ such that $v_{j_h} = v_{j_s}$, and either $(v_{j_{h-1}}, v_{j_h}) \notin R_k$ or $b_{j_{h-1}, j_h} < a_{j_{h-1}, j_h} - 1$. If no such index exists, stop. Otherwise, if $(v_{j_{h-1}}, v_{j_h}) \in R_k$, increment b_{j_{h-1}, j_h} and $b_{j_h, j_{h-1}}$ by 1. Reverse the chain $(v_{j_s}, \ldots, v_{j_h})$ in C_k'.

Step 4: Set $s \leftarrow s + 1$. If $s = t$, go to Step 5, otherwise go to Step 3.

Step 5: Let $C'_k = \left(v_{j_1} = v_{i_r}, \ldots, v_{j_s}, \ldots, v_{j_t} = v_{i_r}\right)$. If $\sum_{h=1}^{s-1} c_{j_h, j_{h+1}} \leq$ the length of the shortest chain between v_{j_1} and v_{j_s}, go to Step 7.

Step 6: Let $P = \left(v_{k_1} = v_{j_1}, \ldots, v_{k_p} = v_{j_s}\right)$ be the shortest chain used in Step 5. Set $C'_k = \left(v_{k_1}, \ldots, v_{k_p}, v_{j_{s+1}}, \ldots, v_{j_t}\right)$, $t \leftarrow t - s + p$, and $r \leftarrow 1$. Go to Step 2.

Step 7: If $r = t - 1$, stop. Otherwise, set $r \leftarrow r + 1$ and go to Step 2.

Given a cycle C_k and a starting vertex indexed by r, steps 2 through 4 of the procedure attempts to find another vertex v_{j_s} such that all the edges of R_k appear at least once in the chain following v_{j_s}. If we identify such a vertex, we can replace the chain from v_{j_1} to v_{j_s} with a shorter chain consisting of non-service edges, if one is available. We repeat this procedure using each vertex in the cycle C_k as the starting vertex of the cycle in an attempt to produce a shorter cycle. We next provide detailed descriptions of procedures DROP and ADD as in Hertz et al. (1996).

Procedure DROP

Given a cycle C_k and a path P on this cycle with at least one service edge, this procedure moves all the service edges on this path from R_k to E, and then applies procedure SHORTEN to the resulting cycle.

Procedure ADD

Given a cycle C_k and a path $P = \left(v_{k_1}, v_{k_2}, \ldots, v_{k_p}\right)$ on another cycle C_l with at least one service edge, the procedure adds this path P to cycle C_k. Thus, $R_k = R_k \cup \{\text{the service edges on path } P\}$.

Step 1: If all the service edges in P are already in R_k, stop.

Step 2: If either v_{k_1} or v_{k_p} (say v_{k_1}) or both vertices appear in C_k, replace $C_k = (v_{j_1}, \ldots, v_{k_1}, \ldots, v_{j_t})$ with $C_k = (v_{j_1}, \ldots, v_{k_1}, v_{k_2}, \ldots, v_{k_p}, \ldots, v_{j_t})$. Go to Step 4.

Step 3: If neither v_{k_1} or v_{k_p} appears in C_k, find the vertex v_{i_r} of C_k yielding $\min_r \{c_{j_r,k_1} + c_{j_r,k_p}\}$. Set $C_k = (v_{j_1}, \ldots, v_{k_r}, v_{k_1}, \ldots, v_{k_p}, v_{k_r}, \ldots, v_{j_t})$.

Step 4: Apply procedure SHORTEN to the cycle C_k.

2.2.5 TABU MOVES

When a path containing a set of service edges is moved from a cycle C_k to another cycle C_l, each one of the edges in the path cannot be moved back from cycle C_l to cycle C_k for θ iterations. In order to determine the value for θ, we use the idea of random tabu tags described in Gendreau et al. (1994). θ is a randomly selected integer in the interval $\theta_{\min} = 5$ and $\theta_{\max} = 10$.

2.2.6 STOPPING RULE

The local search procedure terminates if F_1^* or F_2^* have not decreased for $LS_{\max}$ consecutive iterations. The actual value for $LS_{\max}$ is set to be $|R|$, where R is the set of edges with positive demand, in our implementation. We next provide a detailed description of the tabu search procedure, which is the main part of algorithm TABUCARP.

2.2.7 TABU SEARCH PROCEDURE

The tabu search $Tabu\_Search(S, LS_{\max})$ starts with a given solution S and terminates based on the parameter $LS_{\max}$. We also use a post-optimization procedure that is an adaptation of the Unstringing-Stringing (US) procedure developed by

Gendreau et al. (1992) for the TSP. Hertz et al. (1996) call this adaptation as DROP-ADD and use it for the undirected rural Postman Problem.

Given a cycle C_k, procedure DROP-ADD attempts to find a cycle C_k' serving the same set of edges by successively removing the service edges from the cycle and then adding them at the best possible position. This procedure uses the DROP and ADD procedures described earlier to remove and insert edges into a cycle. The post-optimization procedure DROPs the first service edge from the given cycle and then ADDs this service edge back to the shortened cycle. If the length of the resulting cycle is less than the original cycle, we start again with the first service edge on the resulting cycle. Otherwise, we continue with the remaining service edges until all service edges are dropped and added. We next provide a step by step description of the tabu search procedure.

Step 0: Initialization

Set iteration count $\mu \leftarrow 1$. Consider solution S. Calculate the distance measure and penalty for all cycles. If S is feasible, set $F_1^* \leftarrow F_1(S)$ and $S^* \leftarrow S$. Set $F_2^* \leftarrow F_2(S)$ and $\widetilde{S}^* \leftarrow S$. No move is tabu.

Step 1: Candidate Cycle Selection

Consider solution S and pick the candidate cycle C_k as described in Section 2.2.3.

Step 2: Evaluation of Candidate Moves

- If cycle C_k is an excess cycle perform a drop move, else if it is a deficit cycle, perform an add move, else perform a perturb move. Generate all the neighbors S' of S as described in Section 2.2.4.
- If a neighboring solution S' is generated by a tabu move, it is not considered further, unless S' is feasible and $F_1(S') < F_1^*$, or S' is infeasible and $F_2(S') < F_2^*$.

Step 3: Identification of Best Move

Set $\widetilde{S} \leftarrow \text{argmin } \{F_2(S')\}$

Step 4: Identification of Next Solution

Solution S is set equal to $\widetilde{S}$ unless the following three conditions are satisfied: (i) $F_2(\widetilde{S}) > F_2(S)$, (ii) S is feasible, and (iii) procedure DROP-ADD was not used at iteration μ-1. If the three conditions are satisfied, it means that we have a feasible solution with a lower objective value that was not obtained using the DROP-ADD procedure. So there is a possibility that we might be able to improve the feasible solution S using DROP-ADD. In this case, the new solution is obtained by applying procedure DROP-ADD to S.

Step 5: Update

Set $\mu \leftarrow \mu + 1$. Adjust penalty function as described in Section 2.2.2, if necessary. If procedure US was not used in Step 4, tabu moves. Update distance measures and penalties for all cycles of the new solution, and best known solutions and objective function values.

Step 6: Termination Check

If F_1^* and F_2^* have not decreased for the past LS_{max} iterations, stop. Otherwise go to Step 1.

2.3 TABUCARP ALGORITHM

TABUCARP is motivated by a recent work of Rochat and Taillard (1995) on using probabilistic diversification and intensification strategies for VRPs. The algorithm exploits an adaptive memory that contains individual routes from previously visited best solutions. The routes from several different solutions are then combined to form new starting solutions for the tabu search procedure described in the previous section.

Genetic algorithms (Holland 1975) inspire the concept of combining parts of previous solutions to form a new solution. The procedure starts with an adaptive memory filled with a pool of routes from several different initial solutions. TABUCARP then methodically combines several routes in the adaptive memory to form new starting solutions, applies tabu search to these solutions, and uses the routes from the resulting solutions to update the adaptive memory.

2.3.1 INITIAL SOLUTIONS

In order to start the TABUCARP procedure we have to fill the adaptive memory with routes from several different initial solutions. Given a giant Euler tour over all the edges with positive demand, we generate $\phi = \max\left(|R| - m, 15\right)$ different initial solutions as follows.

Step 0: Set $k \leftarrow 1, l \leftarrow 1$.

Step 1: Construct the kth solution as follows. Start at the depot and take the shortest path to the lth edge on the Euler tour that requires service. The first vehicle services edges starting with this edge and proceeds until its capacity is blocked. If the capacity is violated while servicing edge (v_i, v_j), the vehicle returns to the depot from node v_i. The second vehicle starts from the depot and reaches node v_i. The first edge serviced by this vehicle is edge (v_i, v_j) and the cycle construction proceeds as for the first vehicle. The process continues until either all edges are serviced, or until cycles are constructed for (m - 1) vehicles. In this case, the remaining edges are assigned to vehicle m (the solution may be infeasible in this case).

Step 2: Set $k \leftarrow k+1$. If $k > \phi$, stop. If not, go to Step 3.

Step 3: Set $l \leftarrow l+1$. If the lth service edge on the giant tour is a starting edge for one of the vehicle tours on the first initial solution go back to Step 3. If not, go to Step 1.

2.3.2 ADAPTIVE MEMORY

In the beginning, the adaptive memory contains individual cycles from the initial solutions. All the cycles associated with a single solution are contiguous and the solutions are sorted in the ascending order of their objective function value. The process of selection of cycles for forming a new incumbent solution is biased in favor of the best cycles. If the size of the adaptive memory C is $|C|$, then the cycle in position i has a probability of $(|C|-i+1/\text{sum of all the position indices}) = \left(|C|-i+1\Big/\sum_{j=1}^{|C|}(|C|-j+1)\right)=(2(|C|-i+1))/(|C||C+1|)$. In our implementation, we restrict the size of the adaptive memory to 250 routes. After sorting the cycles in the adaptive memory, if the number of cycles is more than 250, we remove the last $|C|$ - 250 cycles from the adaptive memory. If we let the adaptive memory grow indefinitely, the computational time would increase dramatically. At the same time, too small an adaptive memory would limit the capability of the search process to develop new solutions. Our initial experimentation indicated that the computational times increase dramatically when we increased the size from 250 to 300 and the improvement in the solution quality was marginal.

As Rochat and Taillard explain, once we perform the probabilistic selection of cycles several times, the search tends to concentrate in promising regions of the solution space. This is due to the fact that we choose the cycles with a bias towards "better" cycles and the worst cycles are removed from the adaptive memory to maintain its size. Also note that since we allow identical cycles (from different solutions) to be added to the adaptive memory, the "better" cycles are more often used to form the new incumbent solution and the process slowly changes from a diversification phase to an intensification phase.

2.3.3 DESCRIPTION OF TABUCARP

Step 0: Initialization

- Generate ϕ different initial solutions as described in Section 3.1.
- For each solution S,

  Call procedure $Tabu\_Search(S, |R|)$.

  Label each cycle of the resulting solution with the value of the solution.

  Load cycles with more than one service edge into adaptive memory (defined as set C).

Step 1: Diversification and Intensification

- Repeat the following steps for 100 iterations.
  - Sort the cycles in adaptive memory in ascending order of the labels.
  - Set $C' \leftarrow C$ and $S \leftarrow \varnothing$.
  - While $C' \neq \varnothing$, do the following:
    . Choose a cycle from C', probabilistically such that the cycle in the ith position of C' has a probability of $(2(|C'|-i+1))/|C'||C'+1|$ of being selected.
    . Add this cycle to S and remove from C' all cycles with one or more edges belonging to this cycle.
  - If some of the edges of R are not serviced by the cycles in S, construct an additional cycle with these edges and add to the solution S.
  - Call procedure $Tabu\_Search(S, |R|)$.
  - Label each cycle of the resulting solution with the value of the solution.
  - Load cycles with more than one service edge into adaptive memory.

2.4 COMPUTATIONAL RESULTS

We coded TABUCARP in C and tested it on 23 problem instances used by Pearn (1989) and Hertz, Laporte and Mittaz (1996) to test their algorithms. The best known

solutions for these instances are produced by either the modified Construct-Strike (MCS) heuristic (Pearn 1989) or CARPET. So we present a comparison of our results with those of CARPET and MCS in Table 1.

We use the instance number as in Hertz et al. (1996). The lower bound we use for comparison (LB2) is the lower bound LB2 described in Benavent et al. (1992), except for instance 15 which has a higher lower bound as reported in Hertz et al. (1996). The deviation of a solution from the lower bound is calculated as (Solution value/Lower bound). The average deviation is the average over the 23 instances and the worst deviation gives the worst deviation over the 23 instances.

Table 1 shows that TABUCARP produces solutions with the same value as CARPET in 22 out of the 23 instances. The total distance traversed by the TABUCARP solution for instance 2 is more than the corresponding value for the CARPET solution. However, a closer investigation of the routes shows that the work load is better balanced among the routes in this instance for the TABUCARP solution. In order to measure the quality of a solution with respect to work load balancing, we define the *Measure of Capacity Deviation* (MCD) as follows. Let E_k be the set of edges with positive demand that is serviced on cycle C_k and B be the set of cycles that service at least one edge. For each cycle $C_k \in B$, let ψ_k be the total demand served on that cycle. Then,

$$\psi_k = \sum_{(v_i, v_j) \in E_k} q_{ij} \tag{2.3}$$

and

$$\text{MCD} = \frac{\text{standard deviation of } \{\psi_k : C_k \in B\}}{\text{average of } \{\psi_k : C_k \in B\}} \times 100 \tag{2.4}$$

The benchmark problems vary quite dramatically in their sizes and hence the computational times also vary. The simplest problem takes less than one second on a Sun Sparc work station and the biggest computational time was recorded for problem 25 (3125 seconds). TABUCARP is in general computationally more expensive when compared to CARPET. TABUCARP takes 1.26 times the time taken by CARPET, on average to solve the benchmark problems.

Instance Number	|V|	|E|	MCS	CARPET	TABUCARP	BEST KNOWN	LB2
1	12	22	323	**316**	**316**	316	310
2	12	26	345	**339***	353	339	339
3	12	22	**275***	**275***	**275***	275	275
4	11	19	**287**	**287**	**287**	287	274
5	13	26	386	**377**	**377**	377	376
6	12	22	315	**298**	**298**	298	295
7	12	22	**325**	**325**	**325**	325	312
10	27	46	366	360	360	348	326
11	27	51	346	**311**	**311**	311	277
12	12	25	**275***	**275***	**275***	275	275
13	22	45	406	**395***	**395***	395	395
14	13	23	645	462	462	458	428
15	10	28	**544***	**544***	**544***	544	544**
16	7	21	102	**100***	**100***	100	100
17	7	21	**58***	**58***	**58***	58	58
18	8	28	**127***	129	129	127	127
19	8	28	**91***	**91***	**91***	91	91
20	9	36	**164***	**164***	**164***	164	164
21	11	11	63	**55***	**55***	55	55
22	11	22	123	123	123	121	121
23	11	33	**156***	158	158	156	156
24	11	44	**200***	**200***	**200***	200	200
25	11	55	**233***	235	235	233	233
Average deviation				1.0579	1.0209	1.0227	
Worst deviation				1.507	1.1227	1.1227	
Number of optima				10	11	10	
Number of best				12	17	16	

* Indicates an optimal solution
** Lower bound as used in Hertz et al. (1996)
Numbers in bold indicate a best known solution

Table 1. Computational Results for DeArmon's Problems

In order to better understand the effect of work load balancing on the solution quality, we tested TABUCARP on 50 instances of randomly generated problems and compared the results with the corresponding CARPET solutions. We generated the random problems as follows. We generated grid networks (specified as a x b, where a is the number of rows and b is the number of columns in the grid) of five different sizes – 7x6, 9x8, 10x5, 10x8, and 12x5. An a x b instance has ab vertices and ($2ab - a - b$) edges. The lengths of the edges were generated according to a discrete uniform distribution on $[l_{v\_\min}, l_{v\_\max}]$ ([4,8] in our experimentation) for the vertical edges and on $[l_{h\_\min}, l_{h\_\max}]$ ([5,10] in our experimentation) for the horizontal edges. Some edges were randomly included in R with a probability generated in one of the two intervals [0.2, 0.5] or [0.6, 0.9]. The depot was chosen randomly from the ab vertices. Demands for the edges in R were generated according to a discrete uniform distribution on $[1, d_{\max}]$. ($d_{\max}$ is set to 10 in our experimentation.)

The vehicle capacities were generated as follows. We are given the number of vehicles m (4 for the lower density problems and 5 for the higher density problems) and vehicle filling capacity ω (90%) for each problem instance. The vehicle capacity Q is set equal to $(\text{Total demand over all edges}) / (m\omega)$. For each combination of problem size and problem density, we generated five instances. We obtained solutions using both TABUCARP and CARPET for all 50 instances. Tables 2 and 3 summarize the results for the random instances.

We express the deviation in TABUCARP's solution value (TC) from CARPET's solution value (CP) as ((TC-CP)/CP)×100 and the deviation in CARPET's MCD (CPM) from TABUCARP's MCD (TCM) ((CPM-TCM)/TCM)×100. From both Tables 2 and 3, we can see that on average the total distance traveled for the CARPET solutions is 3.23% lower for the lower density problems and 2.33% lower for the higher density problems. However, the work loads of the cycles are better balanced in the solutions produced by TABUCARP as indicated by the MCD. On an average, the MCD for TABUCARP solutions is 20.98% lower for the lower density problems and 6.20% lower for the higher density problems. Over all the 50 instances, TABUCARP yields solutions whose MCD are

Problem Size	TABU CARP		CAR PET		DEVIA TION%	
	SOLN/LB2	MCD	SOLN/LB2	MCD	SOLN	MCD
7x6	1.3429	12.84	1.2902	14.60	4.24	13.69
9x8	1.2443	10.08	1.2443	10.08	0.00	0.00
10x5	1.3261	21.80	1.2887	22.86	2.87	4.86
10x8	1.2532	12.50	1.1974	22.02	4.65	76.17
12x5	1.2632	14.62	1.2102	16.11	4.38	10.18
				Average	3.23	20.98

Table 2. Results for Lower Density Problems

Problem Size	TABU CARP		CAR PET		DEVIA TION%	
	SOLN/LB2	MCD	SOLN/LB2	MCD	SOLN	MCD
7x6	1.1625	10.98	1.1069	13.60	5.02	23.81
9x8	1.1039	11.20	1.0873	11.80	1.54	5.36
10x5	1.1660	23.01	1.1121	23.43	5.08	1.81
10x8	1.0647	22.29	1.0647	22.29	0.00	0.00
12x5	1.1068	9.74	1.1068	9.74	0.00	0.00
				Average	2.33	6.20

Table 3. Results for Higher Density Problems

on average 13.60% lower than the MCD for the CARPET solutions; however, the solution value is 2.78% higher on average. It is important to note that CARPET does not attempt to balance the work load and that is the reason for the increase in the MCD for CARPET solutions. But, we compare the two solutions to understand the price one has to pay to achieve work load balance in the solutions. Our results clearly indicate that it is definitely advantageous to consider work load balancing as a secondary objective, since the deterioration in objective function value is marginal when compared to the nature of the routes produced by TABUCARP.

We should also note that the solutions produced by TABUCARP would be quite sensitive to the work load balance term in the objective function. If we removed this term from the objective function and considered only the total distance traveled, our solutions would be worse than CARPET solutions since our neighborhoods and basic moves are based on the idea of balancing the total demand among the cycles.

2.5 CONCLUSION

In this chapter, we have presented TABUCARP, a new tabu search based heuristic, for the CARP. This heuristic attempts to balance the work load among the routes while minimizing the cost of traversing all the edges and satisfying demand. The concept of work load balancing is very important in several applications such as mail delivery and meter reading, where the amount of work done by each service delivery unit has to be fairly equal. We have tested TABUCARP on a set of benchmark problems and on several randomly generated problem instances. On the set of benchmark problems, our heuristic performs as well as CARPET, the heuristic that produces the best known solution for 17 out of the 23 benchmark problems for the CARP. The results from the randomly generated problems indicate that TABUCARP in fact produces routes that are better balanced in terms of work load on average, but the total distance traversed tends to be a little more than the corresponding distances in the CARPET solutions. Over the 50 instances that we tested, TABUCARP's solutions are 13.6% better on average in terms of work load balancing, and the total distance traversed is 2.78% higher, when compared to CARPET solutions.

CHAPTER 3

THE STOCHASTIC EULERIAN TOUR PROBLEM

3.1 INTRODUCTION

One of the most common problems in routing is the design of routes for people or vehicles delivering service. Such routing problems are of two types -- node routing and arc routing problems, depending on whether the service request is at a node or on an arc/edge. The underlying problem for most arc routing problems is determining a giant tour that starts and ends at a designated depot, and traverses all edges requiring service at least once. This is the deterministic *Eulerian Tour Problem* (ETP).

A connected graph is Eulerian if there exists a closed walk in the graph containing each edge exactly once. If the given graph is not Eulerian, the first step is to add a least cost set of arcs or edges to the graph to make it Eulerian. This is called the least cost augmentation problem. Edmonds and Johnson (1973) show that this problem can be solved in polynomial time for the undirected CPP using an adaptation of Edmond's blossom algorithm. Thus, given an undirected graph $G=(V,E)$ in which all the edges in $E\left(|E|=n\right)$ require service, we can make it Eulerian in polynomial time. Once we have this Eulerian graph, we can determine the actual Eulerian tour in polynomial time too. Edmonds and Johnson (1973) have described three different algorithms for the ETP on an undirected graph. These are the end-pairing algorithm, the next-node algorithm, and the maze-search algorithm. The ETP is well solved for directed and mixed graphs also. van Aardenne-Ehrenfest and de Bruijn describe the spanning arborescence algorithm for the ETP on directed graphs.

For mixed graphs, one usually assigns directions to the undirected edges to transform the mixed graph into a symmetric graph, and then completely orient the remaining undirected edges so that the indegree equals to the outdegree for all vertices of the graph. Now we can use the spanning arborescence algorithm to determine the Eulerian tour.

It is important to note that there may be more than one Eulerian tour for a given graph. However, all these tours have the same cost and hence there is no optimization involved in the ETP. But there exist quite a few situations in practice, when not all the edges that require service need to be visited everyday. In such cases, the number of edges that require a visit is a random variable. For example, consider a postal carrier who has to deliver mail to n different streets. The postal company wishes to minimize the total walking distance for the carrier. When the carrier has to visit all the n streets every day, any Eulerian tour would suffice, since all the Eulerian tours are of equal length. But in reality, based on the realization of demand, the carrier might have to visit only a subset of the edges requiring service on any particular day.

Consider the following alternative in that situation: the postal carrier follows the predetermined tour as long as he has to visit the next edge on the tour to provide service. If at any point on the tour, the postal carrier does not have to visit an edge, he skips that edge, and takes the shortest path to the next edge on the tour that requires a visit. With this alternative, the ETP takes on a different dimension. The different possible Eulerian tours of a graph yield themselves better to skipping certain edges of the graph. For example consider the Eulerian tours 1 and 2 for the undirected 3x3 grid in Figure 7. All edges in the graph have a weight of 1 and all edges require service. Node **O** represents the depot. The dotted lines represent the edges that are only traversed and not serviced. The numbers on the edges of the two tours represent the order in which one visits the edges in these tours. On a particular day, let us assume that edges **A**, **B**, **C**, and **D** require service. This translates to edges 2, 6, 10, and 14 on tour 1 and edges 4, 5, 12, and 13 on tour 2. If we start at the depot, visit the edges in the same order that they appear in the respective tours and return to the depot

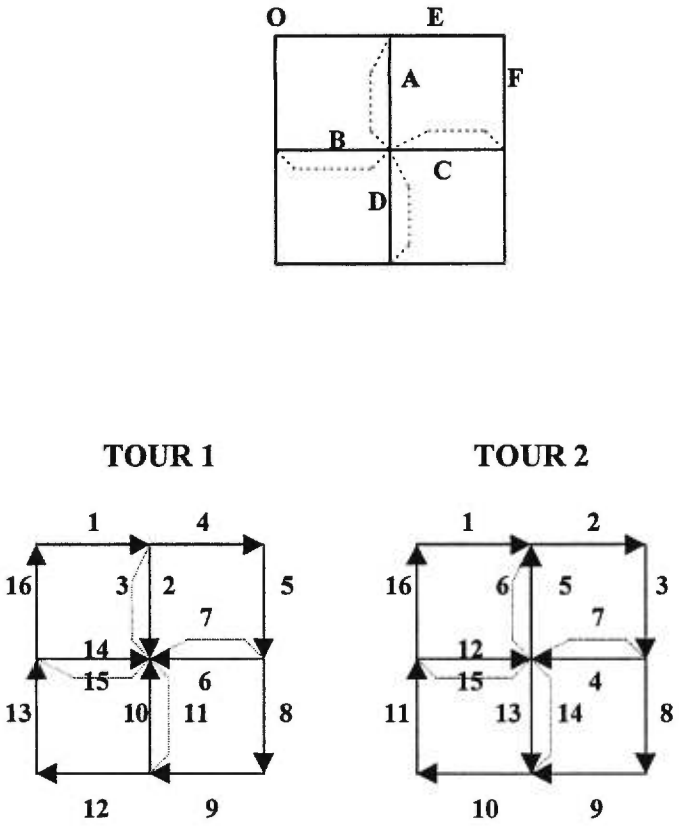

Figure 7. Eulerian graph and two different tours for the same graph

tour 1 results in a length of 10 (Depot-1-**2**-7-**6**-11-**10**-15-**14**-15-16-Depot), while tour 2 results in a length of 12 (Depot-1-2-3-**4**-**5**-6-15-**12**-**13**-10-11-16-Depot). Thus, tour 1 is better for this instance. On the other hand, if edges **A**, **C**, **E** and **F** require service, tour 2 which has a length of 6 (Depot-1-**2**-**3**-**4**-**5**-1-Depot) is better than tour 1 which has a length of 8 (Depot-1-**2**-3-**4**-**5**-**6**-3-1-Depot). Hence, the objective is to determine not just any Eulerian tour, but a particular tour (if more than one tour exists for the given graph) which will have the shortest tour length "on an average". This motivates the investigation of the *Stochastic Eulerian Tour Problem* (SETP) which we define below.

We are given a graph $G=(V,E)$ in which all the edges in $E\,(|E|=n)$ require service (We shall call them "white" edges following the notation in Jaillet (1985).), and a distance $d(v_i,v_j)$ between every pair of directly connected nodes v_i and v_j. On any instance of the problem, only a subset R of the n white edges is present, and hence, requires a visit. The number of present edges follows a specified probability distribution. The objective is to determine an a priori Eulerian tour that visits all the n edges and minimizes the expected length of the tour. On any given instance, one visits and services the present edges in the same order as in the a priori tour, while skipping the ones that are absent. The SETP can be similarly defined when only a subset $R\,(|R|=n)$ of the edges in E consists of white edges. We would like to highlight the fact that we assume that we have solved a least cost augmentation problem for the given graph and have a Eulerian graph and solve the SETP for this Eulerian graph.

Our investigation of the SETP has been motivated by a real-world problem. In the UK postal system, the carriers usually deliver mail a second time in the afternoon. During the first mail delivery, the carriers have to visit all the streets almost always, whereas the second mail delivery is typically very light. Only a small subset of the streets requires service during the afternoon delivery. While any Eulerian tour would be sufficient for the first mail delivery, it is definitely advantageous to determine a tour that minimizes the total length in an expected sense

for the second mail delivery. It is important to note that even though the ETP is well solved, it is not feasible to determine a new tour for each day, since following a new tour every day would decrease the operating efficiency of the postal carrier considerably. In certain applications, like Canada Post, the mail carrier collects the mail to be delivered at various points along the route from relay boxes. On any given day, the present edges are known only after the carrier starts his route and thus, it is not possible for the carrier to determine a new route at the start of each day. In such situations, it is certainly efficient to let the mail carrier follow the same route every day, while allowing the flexibility of skipping streets, if necessary.

Stochastic arc routing is an entirely new area of research. However, researchers have investigated several stochastic node routing problems over the past decade. In the following section, we present some of the related research on stochastic node routing. For results about these studies and a recent survey on stochastic vehicle routing, see Gendreau, Laporte and Séguin (1996). Specifically, Jaillet (1985) introduced the TSP with stochastic customers as the *Probabilistic Traveling Salesman Problem* (PTSP). It is essentially a TSP where each vertex v_i is present with a probability p_i, and hence the number of vertices requiring a visit is a random variable. Jaillet shows that an optimal TSP tour can be arbitrarily bad for the PTSP. He also shows that an optimal tour for the PTSP may intersect itself in the Euclidean plane. This is in contrast to what we know about optimal TSP tours. These results indicate that algorithms have to be developed specifically with the PTSP in mind. Jaillet has developed a number of heuristics by suitably modifying several well-known TSP heuristics such as the Clarke-Wright algorithm and tour merging algorithms. Rossi and Gavioli (1986) present computational results after testing three of Jaillet's heuristics. Bertsimas (1988) and Bertsimas and Howell (1993) have developed a few more heuristics based on probabilistic 2-opt edge exchange, vertex moves within a tour, and space filling curves (Bartholdi and Platzman 1982). Laporte, Louveaux and Mercure (1994) have applied the integer L-shaped method to the stochastic TSP and have solved instances with up to 50 vertices optimally.

In this chapter, we introduce the SETP and present the motivations for studying this problem. Section 3.2 states the definitions and assumptions for the SETP, and presents the method to obtain the expected length of a given tour efficiently. In this section, we also prove that the SETP is NP-hard. We investigate some of the properties of a given tour and derive bounds for the expected length of this given tour in Section 3.3. Section 3.4 highlights some of the desirable properties in an a priori tour using illustrative examples, and Section 3.5 presents the conclusions for this chapter.

3.2 IMPORTANT RESULTS FOR THE SETP

In this section, we first present the basic definitions and assumptions before formally defining the SETP. We then derive an expression for calculating the expected length of a given tour t. We finally show that the SETP is NP-hard even though the ETP is solvable in polynomial time.

3.2.1 DEFINITIONS AND ASSUMPTIONS

$G=(V,E)$ is an undirected graph where V is the set of nodes and E is the set of edges. All the edges in E require service, and hence, denote the set of white edges. Associated with each edge (v_i,v_j) in E is a non-negative real number $d(v_i,v_j)$, the direct distance from node v_i to node v_j. The graph G has a node designated as the depot where the Eulerian tour starts and ends. In order to facilitate the representation and analysis, we duplicate the depot and represent the duplicated node as v_0, which now serves as the depot. The duplicated node v_0 is connected to the original depot by two edges of length 0.

Given an Eulerian tour t, we have an ordering of the nodes and edges, and thus, a direction of traversal (and service) for each of the n edges in E. If we traverse edge e_i from node v_k to v_l, we define v_k as the in-node for edge e_i (v_i^{in})

and v_l as the out-node for edge e_i $\left(v_i^{out}\right)$. Thus, given the in-node and the out-node for each edge in E, we represent an Eulerian tour t as $t=\left(v_0, v_1^{in}, e_1, v_1^{out}, v_2^{in}, e_2, v_2^{out}, \ldots, v_n^{in}, e_n, v_n^{out}, v_0\right)$, where the edges $e_1, e_2, \cdots, e_n$ are numbered in their order of appearance in tour t. The length of the tour t, $L(t)$ is given by:

$$L(t)=\sum_{i=1}^{n} l(e_i)+\sum_{i=0}^{n} d\left(v_i^{out}, v_{i+1}^{in}\right) \tag{3.1}$$

with $v_0^{out}=v_{n+1}^{in}=v_0$, and

$l(e_i)=$ length of edge e_i

If nodes v_i and v_j are not directly connected, then $d\left(v_i, v_j\right)$ is the shortest distance between v_i and v_j.

Each edge e_i in E is present with probability p_i. Thus, for any given instance, the number of white edges present (i.e., requiring a visit) is a random variable. We assume that if k edges require a visit on a particular day, then every set of k edges out of the n white edges is equally likely. Note that when $p_i = p$ for all i, the number of present edges follows a binomial distribution.

Thus, given $G=(V, E)$, a set of n white edges, a distance matrix D and a probability p_i of white edge e_i being present, the SETP seeks an a priori tour that minimizes the expected length of the tour. Looking at it as a stochastic program with recourse, in the first stage, we construct an a priori Eulerian tour of minimum expected length. Once we know the set of present edges, we can describe the second stage solution as follows -- start at the depot, travel to the in-node of the first present edge via the shortest path, traverse and service the first edge and then take the shortest path from the out-node of the first present edge to the in-node of the second present edge. We continue in a similar manner until we reach the out-node of the last present edge and then take the shortest path back to the depot. Given this recourse action and

the precise definition for tour representation, we are ready to present the results for calculating $E[L_t]$, the expected length of a given tour t.

3.2.2 EXPECTED LENGTH OF A GIVEN TOUR

The length of any given tour consists of two parts, namely, the total length of the present white edges, and the total distance traveled from the out-node of one present white edge to the in-node of the next present white edge (i.e., the inter-edge traversal distances). The SETP is similar to the PTSP with n white nodes and one depot in certain aspects. The inter-city traversal distances in the PTSP would correspond to the inter-edge traversal distances. The main difference between the two problems is that in the SETP, in addition to the inter-edge traversal distances, we have to consider the length of the white edges also. Thus, many off our results are extensions of Jaillet's (1985, 1988) results for the PTSP.

In order to derive a concise expression for the inter-edge traversal distances, we define the following n quantities that are similar to the ones defined in Jaillet (1988).

Let $$L_t^r = \sum_{j=0}^{n} d\left(v_j^{out}, v_{\overline{j+r+1}}^{in}\right) \qquad \forall\, r \in \{0,\cdots,n-1\} \qquad (3.2)$$

where $$v_0^{out} = v_{n+1}^{in} = v_0$$

$$\overline{j+r+1} = (j+r) \bmod (n+1) + 1$$

$$d\left(v_j^{out}, v_{\overline{j+r+1}}^{in}\right) = \begin{cases} d\left(v_j^{out}, v_{j+r+1}^{in}\right) & \text{if } 0 \le j \le n-r \\ d\left(v_j^{out}, v_0\right) + d\left(v_0, v_{j-n+r}^{in}\right) & \text{if } n-r < j \le n \end{cases}$$

Note that when $r = 0$, L_t^0 is the total inter-edge traversal distance for the given tour t and $L_t^0 + \sum_{i=1}^{n} l(e_i)$ is the length of the given Eulerian tour t. For $1 \le r \le n-1$, L_t^r is

the sum of $(n+1)$ elements. Each element represents the distance from the out-node of edge e_j to the in-node of its $(r+1)^{th}$ successor edge (i.e., edge $e_{\overline{j+r+1}}$) with respect to the given tour t, i.e., we start at the out-node of edge e_j, skip the next r edges on the tour and travel to the in-node of the $(r+1)^{th}$ edge following edge e_j. Note that when $n-r < j \le n$, to reach the in-node of edge $e_{\overline{j+r+1}}$ from the out-node of edge e_j, we define $d\left(v_j^{out}, v_{\overline{j+r+1}}^{in}\right)$ as reaching the depot from the out-node of e_j and then traveling from the depot to the in-node of $e_{\overline{j+r+1}}$. We first obtain the conditional expected length of a given tour t when k of the n white edges are not present.

Lemma 1: Given a graph G with n white edges, a designated depot v_0, and a probability of occurrence p for each white edge, the conditional expected length of a tour t, given k of the n white edges are not present in the given tour t is

$E\left[L_t \middle| (n-k) \text{ edges present}\right] =$

$$\begin{cases} 0 & \text{if } k = n \\ (1/n)\left[\sum_{i=1}^{n} l(e_i) + L_t^{n-1}\right] & \text{if } k = n-1 \\ \left(1\Big/\binom{n}{k}\right)\left[\sum_{i=1}^{n}\binom{n-1}{k} l(e_i) + \sum_{r=0}^{k}\binom{n-2-r}{k-r} L_t^r\right] & \text{if } k = 0,1,\ldots,n-2 \end{cases} \qquad (3.3)$$

Proof: (i) $k = n$: Since none of the white edges are present, this case is obvious.

(ii) $k = n-1$: Only one white edge is present and each one of the n edges is equally likely to be present. We have to travel from the depot to the in-node of this present edge, service the edge, and travel from its out-node back to the depot. Thus, from the definition of L_t^{n-1}, this case follows.

(iii) $k = 0,1,\ldots,n-2$: The expected length can be described as the sum of the total length of the white edges traversed (the first term in the expression) and the total inter-edge traversal distances (the second term in the expression). The proof for the second term is similar to the proof of Lemma 2.2 of Jaillet (1985) with the following differences. Instead of traveling from a present node to the next present node as in the PTSP, we travel from the out-node of a present edge to the in-node of the next present edge.

Note that we can have k of the n white edges missing in $\binom{n}{k}$ different equiprobable ways. When k edges are missing, the resulting total inter-edge traversal distance is a sum of $n-k$ elements. Since some of the elements might be repeated in the various combinations, we regroup them using L_t^r. Consider an element $d\left(v_j^{out}, v_{\overline{j+r+1}}^{in}\right)$ of L_t^r for a given $r \in \{0,1,\ldots,n-2\}$. For this element to be included, the white edges e_j and $e_{\overline{j+r+1}}$ have to be present and the edges between them have to be absent. Since we have only k edges missing, if $r > k$, $d\left(v_j^{out}, v_{\overline{j+r+1}}^{in}\right)$ will never appear and L_t^r will not be used in calculating $E\left[L_t \middle| (n-k)\text{edges present}\right]$. However, if $r \le k$, we still need to choose the remaining $k-r$ white edges that are missing from a total of $n-2-r$ available white edges, and this can be done in $\binom{n-2-r}{k-r}$ ways. Since this is valid for all $j \in \{0,1,\ldots,n\}$, it follows that each L_t^r appears $\binom{n-2-r}{k-r}$ times while calculating $E\left[L_t \middle| (n-k)\text{edges present}\right]$.

Now, we have to account for the number of times the white edges are actually traversed. When k out of the n edges are missing, we have

$\binom{n}{n-k} = \binom{n}{k}$ different combinations of the present edges. Each combination has $(n-k)$ white edges. Since each one of the n white edges appears an equal number of times over all the combinations, the number of times an edge e_i occurs in $E[L_t]$ is $\binom{n}{k}(n-k)/n = \binom{n-1}{k}$. •

Now that we have a closed form expression for the conditional expected length, we can calculate $E[L_t]$ using the appropriate probabilities.

Theorem 1: Given a graph G, with n white edges, a designated depot v_0, and a probability of occurrence p for each white edge, the expected length of a given tour t is:

$$E[L_t] = p^2\left[\sum_{r=0}^{n-2}(1-p)^r L_t^r\right] + p(1-p)^{n-1} L_t^{n-1} + p\left[\sum_{i=1}^{n} l(e_i)\right] \qquad (3.4)$$

Proof: $E[L_t] = \sum_{k=0}^{n} E[L_t \mid n-k \text{ edges present}] \times \text{Prob}\{n-k \text{ edges present}\}$ (3.5)

where, $\text{Prob}\{n-k \text{ edges present}\} = \binom{n}{k} p^{n-k}(1-p)^k$

$$E[L_t] = \sum_{k=0}^{n-2}\left[\left(1/\binom{n}{k}\right)\left(\sum_{i=1}^{n}\binom{n-1}{k} l(e_i) + \sum_{r=0}^{k}\binom{n-2-r}{k-r} L_t^r\right)\right]\left[\binom{n}{k} p^{n-k}(1-p)^k\right] +$$
$$(1/n)\left[\sum_{i=1}^{n} l(e_i) + L_t^{n-1}\right]\left[\binom{n}{n-1} p(1-p)^{n-1}\right]$$
$$= \sum_{k=0}^{n-2}\left[\sum_{r=0}^{k}\binom{n-2-r}{k-r} L_t^r\, p^{n-k}(1-p)^k\right] + \sum_{k=0}^{n-2}\left[\sum_{i=1}^{n}\binom{n-1}{k} l(e_i)\, p^{n-k}(1-p)^k\right] +$$
$$\sum_{i=1}^{n} l(e_i)\, p(1-p)^{n-1} + L_t^{n-1}\, p(1-p)^{n-1} \qquad (3.6)$$

Let us now consider each term of (3.6).

The first term can be expressed as:

$$\sum_{r=0}^{n-2} L_t^r \left[\sum_{k=r}^{n-2} \binom{n-2-r}{k-r} p^{n-k} (1-p)^k \right] \tag{3.7}$$

Setting $u = k - r$ and $s = n - 2 - r$, (3.7) reduces to

$$\sum_{r=0}^{n-2} p^2 (1-p)^r L_t^r \left[\sum_{u=0}^{s} \binom{s}{u} p^{s-u} (1-p)^u \right] = \sum_{r=0}^{n-2} p^2 (1-p)^r L_t^r \tag{3.8}$$

The second and third terms can be combined as:

$$\sum_{i=1}^{n} l(e_i) \left[\sum_{k=0}^{n-1} \binom{n-1}{k} p^{n-k} (1-p)^k \right]$$

$$= p \sum_{i=1}^{n} l(e_i) \left[\sum_{k=0}^{n-1} \binom{n-1}{k} p^{n-1-k} (1-p)^k \right] = p \sum_{i=1}^{n} l(e_i) \tag{3.9}$$

From (3.6), (3.8), and (3.9), we get

$$E[L_t] = p^2 \left[\sum_{r=0}^{n-2} (1-p)^r L_t^r \right] + p(1-p)^{n-1} L_t^{n-1} + p \left[\sum_{i=1}^{n} l(e_i) \right] \bullet$$

Since each L_t^r is the sum of $n+1$ elements, given a tour t, we can calculate $E[L_t]$ in $O(n^2)$ time using (3.4). Also, note that if the number of white edges present does not follow a binomial distribution, but some other specified probability distribution, we can substitute the appropriate probabilities in (3.5) to calculate $E[L_t]$. However, the $(n-k)$ present edges must be chosen at random from the set of n white edges to use (3.5). In certain scenarios when each white edge e_i is present with probability p_i, we can calculate $E[L_t]$ using the following formula.

$$E[L_t] = \sum_{i=1}^{n} p_i l(e_i) + \sum_{i=1}^{n} d(v_0, v_i^{in}) p_i \prod_{k=1}^{i-1} (1-p_k) + \sum_{i=1}^{n} d(v_i^{out}, v_0) p_i \prod_{k=i+1}^{n} (1-p_k)$$

$$+ \sum_{i=1}^{n} \sum_{j=i+1}^{n} d(v_i^{out}, v_j^{in}) p_i p_j \prod_{k=i+1}^{j-1} (1-p_k) \tag{3.10}$$

We can derive (3.10) by looking at the probability of the following events:

- each white edge being present (first term)
- link between the depot and the in-node of each white edge being present (second term)
- link between the out-node of each white edge and the depot being present (third term)
- link between the out-node of a white edge e_i and the in-node of each white edge following e_i in tour t being present (fourth term)

Expression (3.10) is similar to the closed form expression for calculating the expected length of a given tour for the PTSP with one black node (the depot) and n white nodes. The main difference is that for the SETP, we have to consider the length of the white edges in addition to the distance traveled between edges. We use this similarity to prove that the SETP is NP-hard, in the following section.

3.2.3 NP-COMPLETENESS

We first define the decision problems for the SETP and the PTSP before presenting the NP-completeness proof.

Decision Problem for the SETP

Given an undirected graph $G=(V,E)$, a distance $d(v_i,v_j)$ associated with every edge in E, a subset $R(|R|=n)$ of E consisting of the white edges, a rational number p, $0<p<1$ and a bound B, is there an Eulerian tour t for G with

$$E[L_t]=p^2\sum_{r=0}^{n}(1-p)^r L_t^r + p(1-p)^{n-1}L_t^{n-1} + p\sum_{i=1}^{n} l(e_i) \le B\,?$$

Decision Problem for the PTSP

Given an undirected graph $G' = (V', E')$, $|V'| = n+1$, a distance $d'(v_i, v_j)$ associated with every edge in E', a black node v_0 (the depot) and n white nodes, a rational number q, $0 < q < 1$ and a bound B', is there a traveling salesman tour t' for G' with

$$E[L_{t'}] = q^2 \sum_{r=0}^{n} (1-q)^r L_{t'}^r + q(1-q)^{n-1} L_{t'}^{n-1} \leq B' ?$$

Theorem 2: The SETP is NP-hard.

Proof: The SETP belongs to class NP, since given a tour t, we can calculate $E[L_t]$ in polynomial time $(O(n^2))$ which we can then compare with the bound B. We reduce the PTSP to the SETP to show that it belongs to the class NP-complete.

Given an instance of the PTSP, we construct the graph $G = (V, E)$ as follows:

For each white node v_i in the PTSP, define two nodes v_i^{in} and v_i^{out}.

$$V = \{v_i^{in}, v_i^{out} \ \forall\ i = 1,\ldots,n\} \cup \{v_0\}$$

$$E = \{(v_i^{in}, v_i^{out}), \forall\ i = 1,\ldots,n\} \cup \{(v_0, v_i^{in}), (v_0, v_i^{out}) \forall\ i = 1,\ldots,n\}$$
$$\cup \{(v_i^{out}, v_j^{in}), \forall (v_i, v_j) \in E'\}$$

$$R = \{(v_i^{in}, v_i^{out}), \forall\ i = 1,\ldots,n\}$$

$$d(v_i^{in}, v_i^{out}) = 0, \forall\ i = 1,\ldots,n; d(v_i^{out}, v_j^{in}) = d'(v_i, v_j), \forall (v_i, v_j) \in E',\ i, j = 1,\ldots,n$$

$$d(v_0, v_i^{in}) = d(v_0, v_i^{out}) = d'(v_0, v_i),\ i = 1,\ldots,n$$

$$B = B' \ ;\ p = q$$

Let $t = (v_0, v_1^{in}, e_1, v_1^{out}, v_2^{in}, e_2, v_2^{out}, \ldots, v_n^{in}, e_n, v_n^{out}, v_0)$ be a feasible Eulerian tour for G with $E[L_t] \leq B$. Since each edge e_i is of length 0, it is clear that from the Eulerian tour t, we can construct a tour $t' = (v_0, v_1, \ldots, v_n, v_0)$, which is feasible for

G'. Also, note that $L_t^r = L_{t'}^r$ for all $r = 0,\ldots,n-1$ by construction of G, and hence $E[L_{t'}] = E[L_t] \le B = B'$. Similarly, we can show that if the PTSP has a feasible solution with $E[L_{t'}] \le B'$, then the SETP has a feasible solution with $E[L_t] \le B$, and hence the SETP is NP-hard. •

3.3 PROPERTIES AND BOUNDS FOR THE SETP

In this section, we examine a few properties of the SETP. In order to derive these properties, we express $E[L_t]$ succinctly in a weight-form notation as in Jaillet (1985). Let W be the random variable that represents the number of present white edges.

$$E[L_t] = \sum_{r=0}^{n-1} \alpha_r L_t^r + C \sum_{k=0}^{n-1} \beta_k$$

where, $\alpha_r = \sum_{k=r}^{n-2} \left[\binom{n-2-r}{k-r} \Big/ \binom{n}{k} \right] \text{Prob}(W = n-k) \qquad \forall r \in \{0,\cdots,n-2\}$

$$\alpha_{n-1} = \text{Prob}(W = 1)/n, \quad \beta_k = \left(\frac{n-k}{n} \right) \text{Prob}(W = n-k), \text{ and } C = \sum_{i=1}^{n} l(e_i).$$

We first describe a few properties of α_r, β_k and L_t^r. We then use these properties to derive an expression for the maximum deviation of the expected length of a given Eulerian tour t from the expected length of the optimal Eulerian tour for the SETP, t^*. We should point out that many of the properties associated with α_r and L_t^r are similar to the corresponding properties for the PTSP; however, there are quite a few differences that allow one to derive further simplifications for the SETP. When the properties are different, we explain them in detail; otherwise, we refer the reader to Jaillet (1985, 1988) for detailed explanations.

Property 1: Given a tour t for an Eulerian graph G with n white edges,

L_t^{n-1} is a constant independent of t;

$L_t^1, \ldots, L_t^{n-2}$ are tour-dependent; and

$L_t^0 + C$ is the length of the tour t

From the definition of L_t^r, we see that $L_t^{n-1} = \sum_{i=1}^{n} \left[d\left(v_i^{out}, v_0\right) + d\left(v_0, v_i^{in}\right) \right]$, and hence, it is tour independent. However, $L_t^r, r = 1, \ldots, n-2$ depend on the order in which the edges are visited, and hence are tour independent. By definition, L_t^0 gives the shortest distances between the out-node of the edge e_i to the in-node of edge e_{i+1} for all white edges. Thus, $L_t^0 + \sum_{i=1}^{n} l(e_i)$ is the length of the given tour t. Note that in the PTSP, L_t^0 is the length of the given tour t.

Property 2: The set of edges that define L_t^r along with E, the set of required edges, consists of $(r+1)$ sub-tours, each starting and ending at the depot v_0. This is very similar to the PTSP, except that one needs to consider E, the set of the required edges.

It is important to note that even if the given distance matrix D is symmetric (and hence the matrix of shortest path distances is also symmetric), $d\left(v_i^{out}, v_j^{in}\right)$is in general, not equal to $d\left(v_j^{out}, v_i^{in}\right)$. Hence, several properties of the PTSP do not hold for the SETP. However, if we assume D to satisfy the triangular inequality, then $d\left(v_i^{out}, v_j^{in}\right) \le d\left(v_i^{out}, v_k^{in}\right) + d\left(v_k^{out}, v_j^{in}\right)$, i.e., the inter-edge traversal distances also satisfy the triangular inequality. Under this condition, we can deduce the following.

Property 3: If D satisfies the triangular inequality, we can show that, for a given tour t of an Eulerian graph G,

$$L_t^r \geq L_t^0 \qquad \forall r \in \{0,\ldots,n-1\}$$

This follows since, $L_t^r + C =$ length of $(r+1)$ sub-tours with v_0 as a common node

$\geq$ length of the given Eulerian tour t

$= L_t^0 + C$

From property 2, we know that the set of edges that define L_t^r along with the set of white edges form $(r+1)$ sub-tours with the depot as a common node. Under the triangularity assumption, we can merge these $(r+1)$ sub-tours into a single tour whose length will be less than or equal to the length of the given tour. The length of this merged tour is in turn greater than or equal to the length of the given Eulerian tour t.

Hence, $L_t^r \geq L_t^0 \qquad \forall r \in \{0,\ldots,n-1\}$.

Property 4: Given a tour t of an Eulerian graph G with a depot v_0 and n white edges,

$$L_t^r \leq L_t^{r_1} + L_t^{r-r_1-1} \qquad \forall r \in \{1,\ldots,n-1\}, \text{and } 0 \leq r_1 \leq r-1$$

Hence, $L_t^r \leq (r+1)L_t^0 \qquad \forall r \in \{1,\ldots,n-1\}$

The proof is similar to that of Lemma 3.3 of Jaillet (1985) since only the inter-edge traversal distances are involved. We present here an adaptation of Jaillet's proof to our situation.

$$L_t^r = \sum_{j=0}^{n} d\left(v_j^{out}, v_{j+r+1}^{in}\right)$$

$$\leq \sum_{j=0}^{n} d\left(v_j^{out}, v_{j+r_1+1}^{in}\right) + \sum_{j=0}^{n} d\left(v_{j+r_1+1}^{out}, v_{j+r+1}^{in}\right)$$

by the triangularity assumption

But, $\sum_{j=0}^{n} d\left(v_{j+r_1+1}^{out}, v_{j+r+1}^{in}\right) = \sum_{j=0}^{n} d\left(v_j^{out}, v_{j+r-r_1-1+1}^{in}\right)$

Hence, $L_t^r \le L_t^{r_1} + L_t^{r-r_1-1} \qquad \forall r \in \{1,\ldots,n-1\}$, and $0 \le r_1 \le r-1$

If we let, $r_1 = r-1$ in $L_t^r \le L_t^{r_1} + L_t^{r-r_1-1}$, we get $L_t^r \le L_t^{r-1} + L_t^0$. Using the same relationship for L_t^{r-1}, we get $L_t^r \le L_t^{r-2} + 2L_t^0$. Thus,

$$L_t^r \le (r+1)L_t^0 \qquad \forall r \in \{1,\ldots,n-1\}$$

Property 5: Given a discrete probability distribution for W,

(i) $\sum_{r=0}^{n-1} \alpha_r = E[W]/n \qquad \forall n \ge 1$

(ii) $\sum_{r=0}^{n-1} (r+1)\alpha_r = 1 - \text{Prob}[W=0] \qquad \forall n \ge 1$

(iii) $\sum_{k=0}^{n-1} \beta_k = E[W]/n \qquad \forall n \ge 1$

(iv) $\sum_{k=0}^{n-1} (n/(n-k))\beta_k = 1 - \text{Prob}[W=0] \qquad \forall n \ge 1$

Parts (i) & (ii) are identical to Fact 3.6 of Jaillet (1985). We present a summary of the proofs below.

(i)
$$\sum_{r=0}^{n-2} \alpha_r = \sum_{r=0}^{n-2} \left[\sum_{k=r}^{n-2} \left(\binom{n-2-r}{k-r} \Big/ \binom{n}{k} \right) \text{Prob}(W = n-k) \right]$$

$$= \sum_{k=0}^{n-2} \left(\text{Prob}(W = n-k) \Big/ \binom{n}{k} \right) \sum_{r=0}^{k} \binom{n-2-r}{k-r} \qquad (3.11)$$

But,
$$\sum_{r=0}^{k} \binom{n-2-r}{k-r} = \binom{n-1}{k} \qquad (3.12)$$

From (3.11) and (3.12), we get

$$\sum_{r=0}^{n-2} \alpha_r = \sum_{k=0}^{n-2} \binom{n-1}{k} \Big/ \binom{n}{k} \text{Prob}(W = n-k) \qquad (3.13)$$

By replacing $n-k$ by u in (3.13), we get

$$\sum_{r=0}^{n-2}\alpha_r = \sum_{u=2}^{n}\frac{u}{n}\text{Prob}(W=u) = \frac{1}{n}\left[E[W]-\text{Prob}(W=1)\right]$$

Since $\alpha_{n-1} = \dfrac{\text{Prob}(W=1)}{n}$, the result follows.

(ii) $$\sum_{r=0}^{n-2}(r+1)\alpha_r = \sum_{r=0}^{n-2}\left[\sum_{k=r}^{n-2}\left(\binom{n-2-r}{k-r}\Big/\binom{n}{k}\right)(r+1)\,\text{Prob}(W=n-k)\right]$$

$$= \sum_{k=0}^{n-2}\left(\text{Prob}(W=n-k)\Big/\binom{n}{k}\right)\sum_{r=0}^{k}(r+1)\binom{n-2-r}{k-r}$$

But, $$\sum_{r=0}^{k}(r+1)\binom{n-2-r}{k-r} = \binom{n}{k}$$

Hence, $$\sum_{r=0}^{n-2}(r+1)\alpha_r = \sum_{k=0}^{n-2}\text{Prob}(W=n-k) = 1-\text{Prob}(W=0)-\text{Prob}(W=1)$$

Since $\alpha_{n-1} = \dfrac{\text{Prob}(W=1)}{n}$, the result follows.

(iii) $$\sum_{k=0}^{n-1}\beta_k = \sum_{k=0}^{n-2}\beta_k + \beta_{n-1}$$

$$\sum_{k=0}^{n-2}\beta_k = \sum_{k=0}^{n-2}\left(\frac{n-k}{n}\right)\text{Prob}(W=n-k)$$

$$= \sum_{u=2}^{n}\left(\frac{u}{n}\right)\text{Prob}(W=u)$$

$$= \frac{1}{n}\left[E[W]-\text{Prob}(W=1)\right]$$

Since, $\beta_{n-1} = \text{Prob}[W=1]/n$, the result follows.

(iv) $$\sum_{k=0}^{n-1}\left(n/(n-k)\right)\beta_k = \sum_{k=0}^{n-1}\text{Prob}[W=n-k] = 1-\text{Prob}[W=0]\bullet$$

Lemma 2: Given a graph G with n white edges, a depot v_0, for any given tour t,

(i) $E[L_t] \geq (E[W]/n)\left[L_t^0 + C\right]$

(ii) $E[L_t] \leq (1 - \text{Prob}[W = 0])\left[L_t^0 + C\right]$

Proof:

(i) $$E[L_t] = \sum_{r=0}^{n-1} \alpha_r L_t^r + C\sum_{k=0}^{n-1} \beta_k$$

$$\geq L_t^0 \sum_{r=0}^{n-1} \alpha_r + C\sum_{k=0}^{n-1} \beta_k \qquad \text{[from Prop. 3]}$$

$$= (E[W]/n)\left[L_t^0 + C\right] \qquad \text{[from Prop. 5]}$$

(ii) $$E[L_t] = \sum_{r=0}^{n-1} \alpha_r L_t^r + C\sum_{k=0}^{n-1} \beta_k$$

$$\leq L_t^0 \sum_{r=0}^{n-1} (r+1)\alpha_r + C\sum_{k=0}^{n-1} (n/(n-k))\beta_k \qquad \text{[from Prop. 3 \& 4]}$$

$$= (1 - \text{Prob}[W = 0])\left[L_t^0 + C\right] \qquad \text{[from Prop. 5]} \bullet$$

Next, we derive the worst case ratio for the expected length of a random Eulerian tour when compared to the optimal tour for the SETP.

Theorem 3: Given a graph G with n white edges, and a designated depot, a distance matrix D that satisfies the triangular inequality, the optimal tour t^* for the SETP and a random Eulerian tour t,

$$\left(E[L_t] - E[L_{t^*}]\right)/E[L_{t^*}] \leq (1 - E[W]/n - \text{Prob}(W = 0))/(E[W]/n)$$

Proof: From Lemma 2,

$$E[L_{t^*}] \geq (L_{t^*}^0 + C)[E[W]/n]$$

Since all Eulerian tours are of the same length, $L_t^0 + C = L_{t^*}^0 + C$, and

$$E[L_{t^*}] \geq (L_t^0 + C)[E[W]/n] \tag{3.14}$$

Also, $$E[L_t] - E[L_{t^*}] \leq (L_t^0 + C)[1 - E[W]/n - \text{Prob}[W = 0]] \tag{3.15}$$

Dividing (3.15) by (3.14), we get the desired result. •

Note that when W follows a binomial distribution, with parameter p, $E[W] = np$ and hence Theorem 3 implies $(E[L_t] - E[L_{t^*}])/(E[L_{t^*}]) \leq (1 - p - (1-p)^n)/p$.

3.4 ILLUSTRATIVE EXAMPLE

In this section, we will present an example to investigate the characteristics of different a priori tours for a given graph. Consider the 3x3 grid network given earlier in Figure 7. All vertical edges are of length 1 and all horizontal edges are of length 2. The dotted lines indicate the edges traversed but not serviced on the Eulerian tour t. All edges are white and W is binomial with parameter p, $(0 < p < 1)$. Note, that the length of any tour is 24. Several tours are possible for this graph. Let us consider tours 3 and 4 given in Figure 8 and compare their expected lengths.

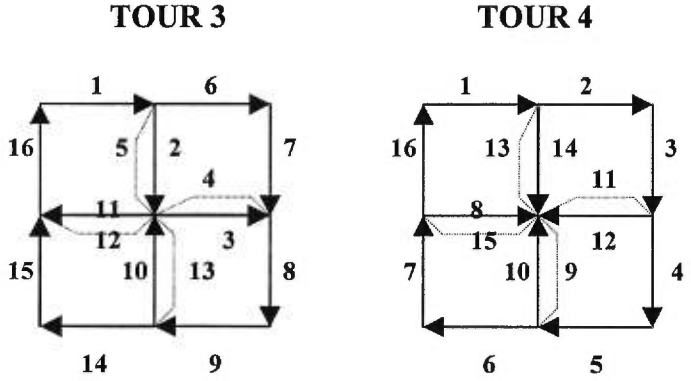

Figure 8. Tours 3 and 4 for the 3x3 grid

The length of both tours is 24. However, the expected length of tour 3, $E[L_{t_3}]$, is less than $E[L_{t_4}]$ for all values of p between 0 and 1. Specifically, for $p = 0.45$, $E[L_{t_3}]$ is

17% lower than $E[L_{t_4}]$. The reason for this is quite obvious from the nature of the tours. In tour 3, edges (2,3,4,5) and edges (10,11,12,13) together form two separate sub-tours. This allows one to skip these sub-tours when the respective edges are not present. Also note that if edges 6 and 7 (or edges 14 and 15) are absent, tour 3 facilitates skipping that section of the tour. On the other hand, tour 4 has only one sub-tour with 8 edges [edges 8-15] other than the outer sub-tour. We can reach the inner sub-tour only after traversing most of the outer sub-tour. Thus, inherently tour 4 necessitates traversing more edges than tour 3, on most instances. Thus, we see that the number of sub-tours clearly has an effect on the expected length of the tour.

Another interesting observation is the effect of the orientation of the edges in the inner sub-tour of tour 4. All the edges are serviced from the outside towards the center. If on a particular instance, edges 2, 8 and 14 are present, the length of the tour is 10. However, if we change just the orientation of edge 14, the length reduces to 8, since, the out-node of edge 8 is the in-node of edge 11 and hence, we do not have to traverse edge 13 before serving edge 14. If we change the orientation of edges 10, 12, and 14, $E[L_{t_4}]$ drops from 20.08 to 19.81 for $p = 0.6$. Hence, another factor to take into consideration while developing tours is the orientation of the edges in a sub-tour.

We also tested other tours to understand the impact of the size of the sub-tours. Consider tour 5 with 2 inner sub-tours (Figure 9). One of the sub-tours has 6 edges (edges 2-7) while the other has only 2 edges (10 and 11).

TOUR 5

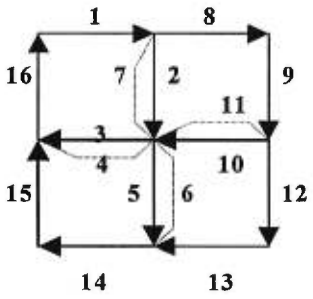

Figure 9. Tour 5 for the 3x3 grid

$E[L_{t_5}]$ is less than $E[L_{t_4}]$ and greater than $E[L_{t_3}]$ for all values of p. Note that tours 5 and 3 have two sub-tours each, while tour 4 has only one sub-tour. Though tours 3 and 5 have the same number of sub-tours, the sub-tours of tour 5 are not balanced.

Our example also illustrates that having a larger number of balanced sub-tours does not necessarily mean lower expected length. Consider the following tour 6 whose edges are oriented exactly the same way as in tour 3. But tour 6 consists of 4 sub-tours each with 2 edges, while tour 3 consists of 2 sub-tours each with 4 edges.

TOUR 6

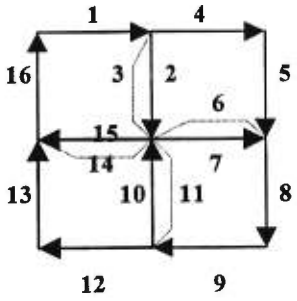

Figure 10. Tour 6 for the 3x3 grid

The expected length of tour 3 is marginally less than that of tour 6 for all values of p. The order in which the edges are visited changes the values of L_t^r for the two tours. Whenever L_t^r for tour 3 is less than the corresponding L_t^r for tour 6, it is for a lower value of r when compared to the opposite situation, and hence the expected length for tour 3 reduces slightly when compared to the expected length of tour 6.

This simple example illustrates that the nature (i.e., number and size) and orientation of the sub-tours play an important role in determining the expected length of the tour. Specifically, the better tours in the expected sense, have more balanced sub-tours when compared to the worse tours. Also, the edges of the sub-tours should be oriented to minimize inter-edge traversal distances.

3.5 CONCLUSION

In this chapter, we first defined the SETP as the problem seeking the Eulerian tour of minimum length in the expected sense, when the number of edges present follow a specified probability distribution. We then derived a closed form expression for calculating the expected length of a given tour in $O(n^2)$ time. We also showed that the SETP is NP-hard even though the deterministic ETP is solvable in polynomial time. We also derived a worst case ratio of the deviation of the expected length of a random Eulerian tour from the optimal tour. Finally, using an illustrative example, we investigated some of the desirable properties in an a priori tour. In the next chapter, we take advantage of the results obtained in this chapter to develop heuristic solution procedures for the SETP.

CHAPTER 4

HEURISTICS FOR THE STOCHASTIC EULERIAN TOUR PROBLEM

4.1 INTRODUCTION

This chapter presents several heuristic procedures for the Stochastic Eulerian Tour Problem (SETP). We defined the SETP in Chapter 3 as follows. We are given an undirected graph $G=(V,E)$ and a distance $d(v_i,v_j)$ between every pair of directly connected nodes v_i and v_j. All of the edges in E $(|E|=n)$ require service and are termed "white edges". On any instance of the problem, only a subset of the n white edges is present, and hence, requires a visit. The number of present edges follows a specified probability distribution. The objective is to determine an a priori Eulerian tour that visits all the n edges and minimizes the expected length of the tour. On any given instance, one visits and services the present edges in the same order as in the a priori tour, while skipping the ones that are absent.

We also presented the UK postal system as a motivation for the investigation of this problem. In this system, the carriers usually deliver mail a second time in the afternoon. During the first mail delivery, the carriers have to visit all the streets almost always, whereas the second mail delivery is typically very light. Only a small subset of the streets requires service during the afternoon delivery. Thus it is definitely advantageous to determine a tour that minimizes the total length in an expected sense for the second mail delivery. It is important to note that even though the deterministic ETP is well solved, it is not feasible to determine a new tour for

each day, since following a new tour each day would decrease the operating efficiency of the postal carrier. In certain applications, like Canada Post, the mail carrier collects the mail to be delivered at various points along the route from relay boxes. On any given day, the present edges are known only after the carrier starts his route and thus, it is not possible for the carrier to determine a new route at the start of each day. In such situations, it is certainly efficient to let the mail carrier follow the same route every day, while allowing the flexibility of skipping streets, if necessary.

In Chapter 3, we also derived a closed form expression for the expected length of a given tour t when the number of present white edges follows a binomial distribution with parameter p. The result can be easily extended when the number of white edges present follows an arbitrary edge invariant discrete probability distribution. We also show that the SETP is NP-hard even though the deterministic Eulerian tour Problem is solvable in polynomial time. We presented several interesting properties of the SETP, and derive bounds for the expected length of a given tour. Finally, the examples presented in that chapter illustrate that the expected lengths of different Eulerian tours vary greatly. Hence it is important to construct tours that would have lower expected length rather than using a random Eulerian tour for any given instance of the problem.

Stochastic arc routing is a relatively new area of research. However, researchers have investigated several stochastic node routing problems over the past decade. Jaillet (1985) introduced the TSP with stochastic customers as the *Probabilistic Traveling Salesman Problem* (PTSP). He derived several theoretical properties of the SETP and has proposed a number of heuristics by suitably modifying several well-known TSP heuristics such as the Clarke-Wright algorithm and nearest neighbor algorithm. Rossi and Gavioli (1986) present computational results after testing three of Jaillet's heuristics. Their computational results indicate that the probabilistic Clarke-Wright algorithm produces TSP tours with lower expected lengths than the corresponding deterministic TSP heuristics when the probabilities are low (between 0 and 0.6). Bertsimas (1988) and Bertsimas and Howell (1993) have developed a few more heuristics based on probabilistic 2-opt edge exchange, vertex

moves within a tour, and space filling curves (Bartholdi and Platzman 1982). Laporte, Louveaux and Mercure (1994) have developed a branch-and-cut algorithm for the stochastic TSP and have solved instances with up to 50 vertices optimally.

As shown in Chapter 3, since the SETP belongs to a class of hard problems, it is not possible to solve realistic sized problems optimally using algorithms that would run in polynomial time. Hence, we concentrate on developing heuristic algorithms that would provide good solutions. In this chapter, we present three different tour construction heuristics for the SETP. The first heuristic is a simple greedy heuristic that determines the next white edge to add to the tour as the one that results in the least increase in the expected length when appended at the end of the tour. The second heuristic also greedily selects the next edge of the tour. The difference is that this heuristic selects the next white edge from the set of edges incident to the current node rather than from the set of all available white edges. Finally the third heuristic constructs several small sub-tours and then concatenates these sub-tours while considering the expected savings in concatenating sub-tours. We have also incorporated an adaptation of a post-optimization procedure introduced by Gendreau, Hertz, and Laporte (1992) for the TSP. Hertz, Laporte, and Nanchen (1996) call this adaptation as DROP-ADD and have used it for the undirected Rural Postman Problem.

We first summarize some of the important results of Chapter 3 in Section 2. This section also presents some of the theoretical preliminaries that we use to design our heuristics. Section 3 presents detailed descriptions of the three tour construction heuristics and the post optimization procedure for the SETP. We present detailed computational results in Section 4 and provide the conclusion and directions for future research in Section 5.

4.2 THEORETICAL PRELIMINARIES

$G=(V,E)$ is an undirected graph where V is the set of nodes and E is the set of edges. Associated with each edge (v_i,v_j) in E is a non-negative real number

$d(v_i, v_j)$, the direct distance from node v_i to node v_j. All the edges in E $(|E| = n)$ are white edges. The graph G has a node designated to be the depot where the Eulerian tour starts and ends. In order to facilitate the representation and analysis, we duplicate the depot and represent the duplicated node as v_0, which now serves as the depot and is connected to the original depot by two edges of length 0, one of which is a white edge.

Given an Eulerian tour t for the graph G, we have an ordering of the nodes and edges, and thus, a direction of traversal (and service) for each of the n white edges. If we traverse edge e_i from node v_k to v_l, we define v_k as the in-node for edge e_i (v_i^{in}) and v_l as the out-node for edge e_i (v_i^{out}). Thus, given the in-node and the out-node for each edge in R, we represent an Eulerian tour t as $t = (v_0, v_1^{in}, e_1, v_1^{out}, v_2^{in}, e_2, v_2^{out}, \ldots, v_n^{in}, e_n, v_n^{out}, v_0)$, where the edges $e_1, e_2, \cdots, e_n$ are numbered in their order of appearance in tour t. When the number of present white edges follows a binomial distribution with parameter p, the expected length of a given tour t can be represented as

$$E[L_t] = p^2 \left[\sum_{r=0}^{n-2} (1-p)^r \, L_t^r \right] + p\,(1-p)^{n-1} \, L_t^{n-1} + p\left[\sum_{i=1}^{n} l(e_i) \right] \qquad (4.1)$$

where $$L_t^r = \sum_{j=0}^{n} d\left(v_j^{out}, v_{\overline{j+r+1}}^{in}\right) \qquad \forall\, r \in \{0, \cdots, n-1\}$$

with $$v_0^{out} = v_{n+1}^{in} = v_0$$

$$\overline{j+r+1} = (j+r) \bmod (n+1) + 1, \text{ and}$$

$$d\left(v_j^{out}, v_{\overline{j+r+1}}^{in}\right) = \begin{cases} d\left(v_j^{out}, v_{j+r+1}^{in}\right) & \text{if } 0 \le j \le n-r \\ d\left(v_j^{out}, v_0\right) + d\left(v_0, v_{j-n+r}^{in}\right) & \text{if } n-r < j \le n \end{cases}$$

Note that if nodes v_i and v_j are not directly connected, then $d(v_i, v_j)$ is the shortest distance between v_i and v_j.

4.2.1 ADDITION OF A WHITE EDGE TO A PATH

Consider a path P starting at the depot v_0 and servicing $i\,(i < n)$ white edges. Let us now add white edge e_{i+1} to the path. We can calculate the expected increase in the length of the path by adding edge e_{i+1} by considering the following two cases:

- the white edges $e_1, e_2, \ldots, e_i$ already in the path are not present;
- k of the i white edges in the path are not present.

Thus, E [increase in length of path by adding edge e_{i+1}] =

$$E\left[I(L_P, e_{i+1})\right] = p\,(1-p)^i\, d\left(v_0, v_{i+1}^{in}\right) + p^2 \sum_{k=0}^{i-1} (1-p)^k\, d\left(v_{i-k}^{out}, v_{i+1}^{in}\right) + p\, l(e_{i+1}) \quad (4.2)$$

The first term of expression (4.2) corresponds to the situation where edge e_{i+1} is the first edge present on the tour and the previous i edges are absent. Hence, we need to consider the direct distance between the depot and the in-node of edge e_{i+1}. The probability associated with this event is $p(1-p)^{i-1}$. Similarly, the second term corresponds to the scenario where exactly k edges $(k = 0,1,2,\ldots,i-1)$ are absent before edge e_{i+1}. In this case, edges e_{i-k} and e_{i+1} are present and edges e_{i-k+1} through e_i are absent. Hence, we need to consider the distance between the out-node of edge e_{i-k} and the in-node of edge e_{i+1}, and the associated probability is $p^2(1-p)^k$. The last term corresponds to the probability that edge e_{i+1} is present.

Note that in order to calculate $E\left[I(L_p, e_{i+1})\right]$ using (4.2), we need to designate one of the ends of edge e_{i+1} as the in-node, i.e., we need to fix v_{i+1}^{in}. If $e_{i+1} = (v_k, v_l)$, we calculate $E\left[I(L_P, e_{i+1})\right]$ for both orientations of e_{i+1}, i.e., by fixing v_{i+1}^{in} as v_k once and as v_l once, and pick the orientation with the minimum expected increase.

4.2.2 MERGING OF SUB-TOURS

The probabilistic Clarke-Wright algorithm proposed by Jaillet (1985) for the PTSP serves as a motivation for the results in this sub-section. In our case, a sub-tour for the given graph G starts at the depot, services $i\,(i < n)$ white edges, and returns to the depot. Let us consider two sub-tours ST_1 and ST_2 which do not service any common white edge. Sub-tours ST_1 and ST_2 service n_1 and n_2 edges, respectively.

$$ST_1 = \left(v_0, v_{1,1}^{in}, e_{1,1}, v_{1,1}^{out}, v_{1,2}^{in}, e_{1,2}, v_{1,2}^{out}, \ldots, v_{1,n_1}^{in}, e_{1,n_1}, v_{1,n_1}^{out}, v_0\right)$$

$$ST_2 = \left(v_0, v_{2,1}^{in}, e_{2,1}, v_{2,1}^{out}, v_{2,2}^{in}, e_{2,2}, v_{2,2}^{out}, \ldots, v_{2,n_2}^{in}, e_{2,n_2}, v_{2,n_2}^{out}, v_0\right)$$

Let $E\left[L_{ST_1}\right]$ and $E\left[L_{ST_2}\right]$ be the expected length of the sub-tours ST_1 and ST_2. When we merge ST_1 and ST_2 by edges e_{1,n_1} and $e_{2,1}$ (i.e., $v_{2,1}^{in}$ directly follows v_{1,n_1}^{out} on the merged sub-tour), the expected length of the merged sub-tour could be less than or equal to the total expected lengths of ST_1 and ST_2. This expected savings in the length of the merged sub-tour could be calculated by considering the following events and the associated probabilities.

- Edge $e_{1,i}\ (1 \le i \le n_1)$ is the last present white edge on ST_1 --

  probability: $p\,(1-p)^{n_1-i}$

- Edge $e_{2,j}\ (1 \le j \le n_2)$ is the first present white edge on ST_2 --

  probability: $(1-p)^{j-1}\,p$

When $e_{1,i}$ is the last present edge on ST_1 and $e_{2,j}$ is the first present edge on ST_2, the savings incurred in the distance traversed is

$$S\left(e_{1,i}, e_{2,j}\right) = d\left(v_{1,i}^{out}, v_0\right) + d\left(v_0, v_{2,j}^{in}\right) - d\left(v_{1,i}^{out}, v_{2,j}^{in}\right) \qquad (4.3)$$

Let $E\left[S\left(\mathrm{ST}_1, ST_2 : e_{1,n_1}, e_{2,1}\right)\right]$ represent the expected savings when we merge ST_1 and ST_2 through edges e_{1,n_1} and $\mathrm{e}_{2,1}$.

$$E\left[S\left(\mathrm{ST}_1, ST_2 : e_{1,n_1}, e_{2,1}\right)\right] = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} S\left(e_{1,i}, e_{2,j}\right) p_{ij} \qquad (4.4)$$

where, $p_{ij} = p^2(1-p)^{(n_1-i)+(j-1)}$

and $S\left(e_{1,i}, e_{2,j}\right)$ is given by (4.3)

Note that we have four different ways to merge ST_1 and ST_2. We can merge the sub-tours through edges e_{1,n_1} and $e_{2,1}$, or e_{1,n_1} and e_{2,n_2}, or $e_{1,1}$ and $e_{2,1}$, or $e_{1,1}$ and e_{2,n_2}. The best way to merge ST_1 and ST_2 is through the edge concatenation that yields the maximum expected savings. Thus, the expected savings from merging ST_1 and ST_2 is

$$E\left[S\left(\mathrm{ST}_1, ST_2\right)\right] = \max_{h \in \{1, n_1\}, k \in \{1, n_2\}} \left[E\left[S\left(ST_1, ST_2 : e_{1,h}, e_{2,k}\right)\right]\right] \qquad (4.5)$$

4.3 HEURISTIC PROCEDURES

In this section, we present detailed descriptions of the three heuristics that we have developed for the SETP. All three are tour construction procedures. The first and the second heuristics construct the tour by adding one white edge at a time, while the third heuristic constructs several small sub-tours and then concatenates these to form the Eulerian tour. Finally, we also describe the post-optimization procedure DROP-ADD that is analogous to the US procedure (Gendreau et al., 1992) for the TSP.

4.3.1 HEURISTIC 1: GLOBAL GREEDY

This heuristic starts with an empty path and successively adds one edge at a time to the path. The edge added to the path is the one that results in the least expected increase in the length of the resulting path. Once all the n edges are added,

we complete the tour by returning to the depot from the out-node of the last added white edge.

Two components contribute to the expected increase in the path length as computed by (4.2). The first is the inter-edge traversal distances between the white edges on the path and the candidate edge, and the second is the length of the candidate edge. Hence, the heuristic tends to select shorter edges closer to the out-node of the last edge in the path. As a result, in certain situations, the tours produced by this heuristic could be longer than necessary when all the white edges are present. For example, consider the graph in Figure 11 with 12 white edges. Node 1 is the depot. The length of all the edges on the three outer triangles is 5 and the length of the edges on the connecting inner triangle is 1.

When each edge in the graph is present with a probability $p = 1.0$, the global greedy heuristic produces the tour (1-4-9-1-2-3-1-4-5-6-4-9-7-8-9-1) of length 51. But, it is obvious that we can reduce the total length to 48 since the edges (1,4), (4,9), and (9,1) are traversed twice. In this case, the expected length will also be reduced since $p = 1.0$.

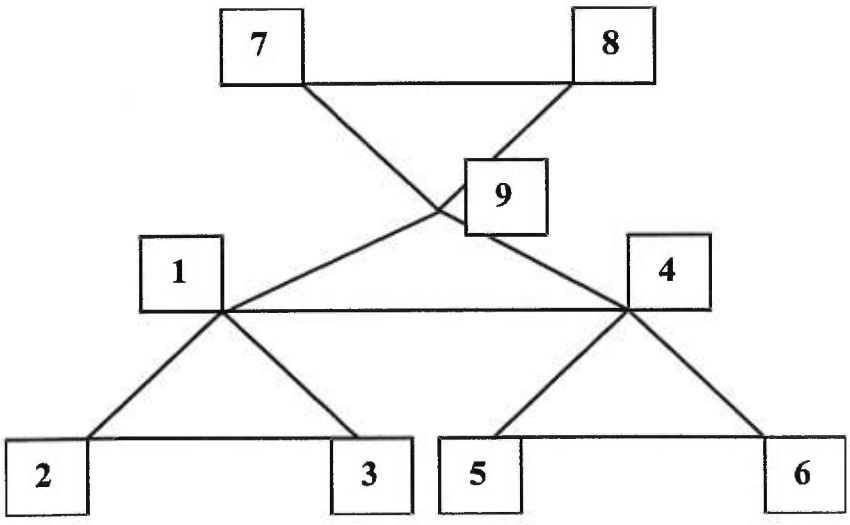

Figure 11. Example graph for heuristic 1

Using this as a motivation, we apply the procedure SHORTEN (Hertz et al., 1996) to the tour produced by the global greedy heuristic. We provide a detailed description of this procedure in Section 2.2.4 of Chapter 2. Basically, this procedure starts with node r (= 1), and moves all the white edges as far to the right as possible. It then replaces the path from node r to the in-node of the first white edge by the shortest chain. If the length of the resulting tour is smaller, we renumber all the nodes on the new tour and start the procedure again at node r (= 1). If not, we increment r by 1 and continue until r is greater than the total number of nodes on the tour.

If we apply procedure SHORTEN to the above tour, when r = 1, the value of the index s is 4, since edges (1,4), (4,9), and (9,1) can be serviced the second time they are traversed. Now we can remove the path from node 1 to node 4 containing only non-service edges and we get the tour (1-2-3-1-4-5-6-4-9-7-8-9-1) of length 48. For values of p less than 1.0, the expected length of the shortened tour could be greater than the expected length of the original tour since the ordering and the orientations of the white edges could be changed during the SHORTEN procedure. Hence, we calculate the expected length of the shortened tour and retain this tour as the final result only if its expected length is less than or equal to the expected length of the original tour. We now present a step-by-step description of algorithm GLOBAL GREEDY.

Step 0: Initialize $P \leftarrow (v_0)$ and $B \leftarrow E$.

Step 1: For every edge e_i in B, calculate $E[I(L_P, e_i)]$ as described in Section 4.2.1.

Step 2: Set $k \leftarrow \text{argmin}\{E[I(L_P, e_i)]\}$. Append the tour segment $(v_k^{in}, e_k, v_k^{out})$ to P. Set $B \leftarrow B - \{e_i\}$.

Step 3: If $B = \varnothing$, append the return path from the out-node of the last added white edge to the depot to P to get tour t and go to Step 4. If not, go to Step 1.

Step 4: Apply procedure SHORTEN to tour t to obtain tour t'. If $E[L_{t'}] \leq E[L_t]$, t' is the final tour, else t is the final tour.

4.3.2 HEURISTIC 2: LOCAL GREEDY

This heuristic is a modification of the global greedy heuristic. At each iteration, instead of selecting the next white edge from among all the available white edges, we choose from only among the edges incident to the node we are at. Thus, this heuristic starts at the depot, adds the white edge with the least expected increase in length from among all the white edges incident to the depot, and repeats the process at the out-node of the added white edge. If at any point, there are no white edges incident to the node we are at, there may be a matching edge incident to this node, that was added to the graph while solving the augmentation problem. If this is the case, we traverse the matching edge and continue the process, until all white edges are added and we have a complete tour.

This heuristic ensures that when $p = 1.0$, the length of the tour generated using this heuristic is equal to the length of a random Eulerian tour. We feel that this heuristic should perform better than the global greedy heuristic for most values of p. The detailed description of the local greedy heuristic is as follows.

Step 0: Initialize $P \leftarrow (v_0)$ and $B \leftarrow E$. Calculate the degree of all the nodes in the graph. Let *current_node* $\leftarrow v_0$.

Step 1: Let W = set of white edges incident to *current_node*. If $W \neq \varnothing$, calculate $E[I(L_P, e_i)]$ as described in Section 4.2.1 for every edge e_i in W, and go to Step 2. If $W = \varnothing$, go to Step 3.

Step 2: Set $k \leftarrow \underset{e_i \in W}{\text{argmin}} \{E[I(L_P, e_i)]\}$. Append the tour segment $(v_k^{in}, e_k, v_k^{out})$ to P. Set $B \leftarrow B - \{e_i\}$. Decrease degree of *current_node* and v_k^{out} by 1. Set *current_node* $\leftarrow v_k^{out}$, and go to Step 4.

Step 3: If degree[*current_node*] > 0, there is a matching edge incident to *current_node*. Traverse this matching edge. Decrease degree of

current_node and the out-node of the matching edge by 1. Set *current_node* ← out_node of the matching edge, and go to Step 4.

If degree[*current_node*] = 0, backtrack on the partial tour just developed, to a node v_i with degree > 0. Set *current_node* ← v_i, and go to step 4.

Step 4: If $B = \varnothing$, append the return path from the out-node of the last added white edge to the depot to P to get tour t and go to Step 5. If not, go to Step 1.

Step 5: Apply procedure SHORTEN to tour t to obtain tour t'. If $E[L_{t'}] \leq E[L_t]$, t' is the final tour, else t is the final tour.

4.3.3 HEURISTIC 3: SUB-TOUR CONSTRUCTION

The third heuristic is also a tour construction heuristic, but is quite different from the global and local greedy heuristics. This heuristic first constructs a single giant sub-tour using a set C of eligible edges, and another outer tour using the edges in $E \backslash C$. The sub-tour construction heuristic then breaks up the giant sub-tour into as many small separate sub-tours as possible. Finally, the heuristic inserts these small sub-tours at appropriate insertion points on the outer tour to obtain the Eulerian tour. We first describe each of the procedures of the algorithm and then provide a detailed description of the overall heuristic.

Determination of the set of eligible edges

This procedure forms a set of edges that can be used to form the giant sub-tour. If the degree of a node is greater than 2, then this node occurs more than once in the final Eulerian tour, and hence there is a sub-tour out of this node. This fact motivates the idea behind forming the set C. In order to determine C, we need to consider only edges whose both end points are of degree greater than 2.

Step 1: Set $C \leftarrow \varnothing$. Calculate the degree of all the nodes in the graph.

Step 2: If the two end points of an edge $e_i \in E$ are both of degree > 2, then add edge $e_i \in E$ to the set C.

Breaking up a single sub-tour into smaller sub-tours

This procedure breaks a sub-tour that starts at the in-node of a white edge in C and ends at the out-node of another white edge in C into as many small sub-tours as possible. Given a giant sub-tour, we have an ordering of the nodes. The procedure uses this ordering and results in more than one sub-tour if one or more nodes are visited more than once in the giant sub-tour. Let the given sub-tour containing k nodes be represented as $ST = (v_{s,1}, v_{s,2}, \ldots, v_{s,k})$. Since the in-node of the first edge on a given sub-tour and the out-node of the last white edge are the same, we do not store the out-node of the last white edge in ST for notational simplicity. However, when we merge a given sub-tour with another sub-tour or the outer tour, we always ensure that the last white edge on the given sub-tour is traversed, by visiting the out-node of the last white edge. We store the resulting smaller sub-tours in the array *sub_tour*.

Step 0: Set $sub\_tour[1] \leftarrow ST$. For each node $v_i \in V$, set $count[v_i] \leftarrow 0$.

Step 1: Set $count[v_{s,i}] \leftarrow count[v_{s,i}] + 1$ for all $i = 1, \ldots, k$, $num\_tours \leftarrow 1$, and $i \leftarrow 1$.

Step 2: If $count[v_{s,i}] > 1$, let j be the position of node $v_{s,i}$ the second time it occurs on the given sub-tour. If not, set $i \leftarrow i + 1$, and go to Step 4.

Step 3: Remove the sub-tour $(v_{s,i}, v_{s,i+1}, \ldots, v_{s,j-1})$ from *ST*. For each node in this smaller sub-tour, decrease count by 1. Set *num_tours* ← *num_tours* + 1, and $sub\_tour[num\_tours] \leftarrow (v_{s,i}, v_{s,i+1}, \ldots, v_{s,j-1})$.

Step 4: If $i < k$, go to Step 2.

If not, if *num_tours* > 1, use this procedure to break up $sub\_tour[2]$ to $sub\_tour[num\_tours]$. If *num_tours* = 1, the given tour has no sub-tours, STOP.

Inserting the sub-tours into the outer tour

This procedure inserts the *num_tours* sub-tours into an outer tour. Each sub-tour ST_i is represented as $ST_i = (v_{s_i,1}, v_{s_i,2}, \ldots, v_{s_i,k_i})$. This procedure determines a common node between the outer tour and each one of the sub-tours, if one exists, and inserts the sub-tours starting at this common node. Note that there is at least one sub-tour having a common node with the outer tour during the first iteration. While inserting the sub-tours, we make sure that we insert the sub-tour directly into the outer tour and not into another sub-tour. We repeat the procedure if there are still some sub-tours remaining to be inserted into the outer tour. We present below a detailed description of the procedure. Note that we store the position on the outer tour at which a common node exists between sub-tour ST_i and the outer tour in $pos\_outer[i]$ and the corresponding position on the sub-tour ST_i in $pos\_subtour[i]$.

Step 0: Let $t_{outer} \leftarrow$ outer tour, and $A \leftarrow$ set containing the *num_tours* sub-tours. Set $pos\_outer[i] \leftarrow 0$ for $i = 1, \ldots, num\_tours$.

Step 1: For $i = 1, \ldots, num\_tours$, if a common node exists between the outer tour and sub-tour ST_i, determine its position on t_{outer} and ST_i, and update $pos\_outer[i]$ and $pos\_subtour[i]$.

Step 2: Let B be the set of all sub-tours with $pos\_outer[i] > 0$, and $|B| \leftarrow m$. Set $j \leftarrow 1$.

Step 3: Let $k \leftarrow \underset{ST_i \in B}{\text{argmax}} \{pos\_outer(i)\}$. Insert sub-tour ST_k starting at the node in $pos\_subtour[k]$ into the outer tour starting at $pos\_outer[k]$. Set $j \leftarrow j+1$, $pos\_outer[k] \leftarrow -1$.

Step 4: If $j \leq m$, go to step 3. If not, go to Step 5.

Step 5: If there is one or more sub-tour ST_i with $pos\_outer[i] = 0$, go to Step 2. If not, t_{outer} contains the final Eulerian tour, stop.

Overall Procedure

Step 1: Construct the set C of the eligible edges for sub-tour construction using the procedure described earlier.

Step 2: For each edge $e_i \in C$, construct a sub-tour $ST_i = (v_0, v_i^{in}, e_i, v_i^{out}, v_0)$. Let A $\leftarrow$ set of all sub-tours.

Step 3: Calculate $E[S(ST_i, ST_j)]$ using (4.5) as described in Section 4.3.2 for every pair of sub-tours in A.

Step 4: Let $k \leftarrow arg\,max\{E[S(ST_i, ST_j)]\}$ and i' and j' be the indices yielding k.

Step 5: Concatenate sub-tours i' and j' using the appropriate orientation of the sub-tours. Add the concatenated tour to A. Set $A \leftarrow A - \{ST_{i'}, ST_{j'}$, and $|A| \leftarrow |A| - 1$.

Step 6: If $|A| = 1$, SHORTEN the giant sub-tour in set A and go to Step 7. If not, go to Step 3.

Step 7: Break up the giant subtour in set A into as many smaller sub-tours as possible using the procedure described earlier.

Step 8: If $E \setminus C \neq \varnothing$, construct a tour t_{outer} with the edges in $E \setminus C$ using the global greedy heuristic and SHORTEN the tour. If not, set $t_{outer} \leftarrow \{v_0\}$.

Step 9: Insert the smaller sub-tours from Step 7 into the outer tour from Step 8 using the procedure described above.

4.3.4 POST-OPTIMIZATION: DROP-ADD

The post-optimization procedure that we use is an adaptation of the Unstringing-Stringing (US) procedure developed by Gendreau et al. (1992) for the TSP. Hertz et al. (1996) call this adaptation as DROP-ADD and use it for the undirected rural Postman Problem. Given an Eulerian tour t with expected length $E[L_t]$, procedure DROP-ADD attempts to find a tour t' with $E[L_{t'}] \leq E[L_t]$ by successively removing

white edges from the solution and then adding them at the best possible position. This procedure uses the ADD procedure as described in Hertz et al. (1996) to insert edges into a tour. We provide a step-by-step description of the DROP-ADD procedure below.

Step 1: Let the given Eulerian tour be t with expected length $z^* = E[L_t]$. The n white edges of the tour are numbered in their order of appearance in the tour. Let $i \leftarrow 1$.

Step 2: Remove edge e_i from the tour t and SHORTEN the tour to obtain tour $\tilde{t}$.

Step 3: Add edge e_i to tour $\tilde{t}$ using procedure ADD as described in Hertz et al. (1996). We explain this procedure in detail in Section 2.2.4 of Chapter 2.

Step 4: Set $t \leftarrow \tilde{t}$. If $E[L_t] < z^*$, set $t' \leftarrow t$, $z^* \leftarrow E[L_t]$, $i \leftarrow 1$; go to Step 2.

Step 5: If $i = n$, stop. If not, set $i \leftarrow i + 1$ and go to Step 2.

4.4 COMPUTATIONAL RESULTS

We coded all the heuristics and the post-optimization procedure in C and tested them on two classes on randomly generated problems. The first class of problems consists of grid networks of varying sizes. We generated grids of sizes 4x4, 5x5, 6x6, 7x7, 8x8, and 9x9 for our computations. For each one of the problems the lengths of the horizontal edges is randomly selected in the interval [5,10] and the length of the vertical edges in the interval [4,8]. All the edges of the grid are white edges. The location of the depot is randomly selected from all the vertices. For each of the grid sizes, we generated 10 instances, thus generating 60 grid networks in all.

For the second class of problems, we generated a specified number of vertices (8, 10, 15, and 20 in our computations) in the $[0,10]^2$ square. We generate a first set of edges by constructing a random Hamiltonian cycle on these vertices. This ensures

that the graph is connected. We then add more edges to the graph randomly until a pre-specified graph density was reached. For our computations, we generate graphs of density 0.3, 0.5 and 0.7 for each value of the number of vertices. We chose the depot as the median of all the vertices of the graph. Finally, for each combination of number of vertices and graph density, we generated 10 problem instances, thus generating 120 Euclidean graph instances in all.

The biggest of the grid networks contains 174 white edges and the biggest Euclidean network contains 153 graphs. In a real world scenario, such as a mail delivery or a meter reading application, the SETP has to be solved for each mail carrier or meter reader separately. Under this condition, we feel that the problem sizes that we have considered are quite realistic.

For all the 180 problem instances, we obtained Eulerian tours when the probability of occurrence of a white edge, p, ranges from 0.1 to 1.0 (in steps of 0.1). For each instance, we also generated a random Eulerian tour, and calculated the expected length of this tour for the different values of p. Tables 4-9 present the results for the grid networks, and Tables 10-21 present the results for the Euclidean networks. Each cell in the tables contains two numbers. The first number represents the average over 10 instances of the ratio of the expected length of the tour obtained using a particular heuristic and the expected length of the random Eulerian tour. The second number gives the average over 10 instances of the time taken in seconds for that heuristic on a Sun Sparc work station.

Each row of the tables presents the average results over 10 instances for a given probability, for all three heuristics without and with the post-optimization procedure. The number in bold (for each row) indicates the heuristic with the best average result over 10 instances for that probability. For example, in Table 4, for a probability of 0.1 the first heuristic along with the DROP-ADD procedure has the best average result over 10 instances. The last row of the tables present the best result over all the 100 runs (10 instances X 10 probabilities of occurrence) for each heuristic without and with the DROP-ADD procedure. The following describes the contents of each column in the tables.

Column 1: Value of p

Column 2: Expected length of tour by global greedy heuristic/expected length of random Eulerian tour

Column 3: Expected length of tour by global greedy heuristic + DROP-ADD/ expected length of random Eulerian tour

Column 4: Expected length of tour by local greedy heuristic/expected length of random Eulerian tour

Column 5: Expected length of tour by local greedy heuristic + DROP-ADD/ expected length of random Eulerian tour

Column 6: Expected length of tour by sub-tour construction heuristic / expected length of random Eulerian tour

Column 7: Expected length of tour by sub-tour construction heuristic + DROP-ADD / expected length of random Eulerian tour

4.4.1 EFFECT OF PROCEDURE DROP-ADD

Our computational results indicate that the post-optimization procedure DROP-ADD is quite effective in producing new Eulerian tours with lower expected lengths. For the grid networks, the expected length of the tours produced by the global and local greedy heuristics drops by 2-4% on average (for the various values of p) after using the DROP-ADD post-optimization procedure. This decrease in the expected length is a little higher for the sub-tour construction heuristic. On a tour produced by the sub-tour construction heuristic, when a white edge is removed from the tour and has to be re-inserted during the DROP-ADD phase, there is typically more options for points of insertion of this edge into the tour. Hence, the post-optimization procedure is more effective on tours produced by the third heuristic. The improvement is the least on tours produced by the second heuristic. For the Euclidean networks, the decrease in the expected length is between 1% and 3% for all three heuristics. The tours produced by the three heuristics for the Euclidean networks are quite similar and hence the effect of DROP-ADD is similar too.

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0079 0.01	**0.9799** 0.24	1.0051 0.00	0.9917 0.08	1.0133 0.01	0.9868 0.12
0.2	1.0128 0.01	**0.9524** 0.20	1.0108 0.00	0.9799 0.09	1.0200 0.01	0.9735 0.11
0.3	1.0142 0.01	**0.9543** 0.15	1.0130 0.00	0.9725 0.08	1.0097 0.01	0.9707 0.10
0.4	1.0039 0.01	0.9570 0.17	1.0133 0.00	0.9703 0.09	1.0055 0.01	**0.9514** 0.11
0.5	1.0485 0.01	0.9768 0.13	1.0131 0.00	**0.9673** 0.08	1.0233 0.01	0.9676 0.09
0.6	1.0761 0.01	**0.9848** 0.18	1.0123 0.00	0.9718 0.09	1.0282 0.01	**0.9619** 0.10
0.7	1.0759 0.01	1.0014 0.13	1.0105 0.00	0.9779 0.09	1.0243 0.01	**0.9690** 0.10
0.8	1.0831 0.01	1.0015 0.15	1.0077 0.00	0.9847 0.08	1.0200 0.01	**0.9787** 0.10
0.9	1.0872 0.01	0.9935 0.12	1.0040 0.00	0.9921 0.08	1.0172 0.01	**0.9878** 0.10
1.0	1.0722 0.00	**1.0000** 0.09	**1.0000** 0.00	**1.0000** 0.06	1.0058 0.01	**1.0000** 0.06
Best	0.92	0.89	0.91	0.88	0.93	0.88

Table 4. Results for the 4x4 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9506 0.04	**0.9395** 1.04	0.9836 0.00	0.9551 0.74	1.0045 0.07	0.9542 0.78
0.2	0.9524 0.03	0.9117 0.81	0.9767 0.00	0.9341 0.77	0.9776 0.08	**0.9074** 0.79
0.3	0.9916 0.03	0.9117 0.69	0.9793 0.00	0.9305 0.72	1.0123 0.08	**0.8939** 0.94
0.4	1.0385 0.04	0.9412 0.75	0.9856 0.00	0.9419 0.70	0.9846 0.08	**0.9280** 0.89
0.5	1.0383 0.03	0.9465 0.95	0.9922 0.00	0.9515 0.68	1.0150 0.08	**0.9437** 0.85
0.6	1.0401 0.03	0.9565 0.78	0.9973 0.00	0.9600 0.68	1.0243 0.08	**0.9466** 0.93
0.7	1.0458 0.03	0.9755 0.78	1.0004 0.01	0.9717 0.66	1.0315 0.08	**0.9583** 0.70
0.8	1.0632 0.03	0.9832 0.75	1.0016 0.00	0.9804 0.74	1.0179 0.10	**0.9657** 0.79
0.9	1.0635 0.03	0.9892 0.81	1.0012 0.00	0.9905 0.71	1.0075 0.10	**0.9796** 0.77
1.0	1.0473 0.02	1.0000 0.40	1.0000 0.00	1.0000 0.26	1.0307 0.06	**0.9956** 0.41
Best	0.91	0.85	0.84	0.77	0.84	0.77

Table 5. Results for the 5x5 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9188 0.11	**0.9165** 3.91	0.9828 0.01	0.9712 0.77	1.0048 0.33	0.9500 2.62
0.2	0.9329 0.11	**0.9143** 3.02	0.9680 0.01	0.9534 0.85	0.9918 0.34	0.9232 2.38
0.3	1.0003 0.11	0.9283 3.23	0.9691 0.00	0.9546 0.88	0.9430 0.37	**0.8852** 2.51
0.4	1.0106 0.11	0.9479 2.68	0.9776 0.00	0.9622 0.89	0.9796 0.38	**0.9179** 2.28
0.5	1.0452 0.11	0.9667 2.29	0.9865 0.00	0.9720 0.89	1.0066 0.39	**0.9441** 1.98
0.6	1.0545 0.11	0.9683 2.84	0.9931 0.00	0.9805 0.88	1.0251 0.39	**0.9605** 2.44
0.7	1.0629 0.11	0.9809 3.05	0.9969 0.00	0.9869 0.89	1.0326 0.40	**0.9787** 2.01
0.8	1.0607 0.11	0.9946 3.23	0.9986 0.01	0.9894 0.94	1.0311 0.40	**0.9892** 2.05
0.9	1.0646 0.11	1.0022 3.09	0.9992 0.01	**0.9945** 0.96	1.0296 0.41	0.9952 1.54
1.0	1.0646 0.06	1.0191 0.89	1.0000 0.00	1.0000 0.63	1.0084 0.25	**0.9962** 0.78
Best	0.86	0.86	0.91	0.88	0.88	0.84

Table 6. Results for the 6x6 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.8778 0.31	**0.8680** 9.55	0.9543 0.02	0.9206 4.88	0.9746 1.02	0.9335 5.81
0.2	0.9224 0.31	0.9043 8.43	0.9399 0.02	0.9056 4.91	0.9684 1.10	**0.8956** 7.93
0.3	0.9907 0.30	0.9145 8.18	0.9492 0.02	0.9174 4.74	0.9760 1.14	**0.8942** 7.94
0.4	0.9835 0.30	0.9388 6.53	0.9654 0.02	0.9351 5.29	0.9852 1.17	**0.9027** 9.69
0.5	1.0172 0.31	0.9632 6.73	0.9805 0.02	0.9520 5.70	1.0095 1.21	**0.9264** 10.27
0.6	1.0669 0.30	0.9750 7.84	0.9915 0.02	0.9674 6.21	1.0279 1.24	**0.9558** 9.64
0.7	1.0517 0.29	0.9882 6.30	0.9982 0.02	0.9795 6.51	1.0462 1.26	**0.9761** 9.61
0.8	1.0482 0.29	0.9853 8.16	1.0011 0.02	0.9891 5.97	1.0624 1.29	**0.9788** 9.04
0.9	1.0530 0.30	0.9929 8.97	1.0013 0.02	**0.9952** 6.04	1.0354 1.31	0.9953 8.28
1.0	1.0597 0.16	1.0053 3.37	**1.0000** 0.02	**1.0000** 1.73	1.0444 0.78	1.0004 3.36
Best	0.79	0.79	0.88	0.83	0.86	0.79

Table 7. Results for the 7x7 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.8801	**0.8643**	0.9337	0.9216	1.0118	0.9060
	0.72	22.05	0.03	6.13	2.61	19.55
0.2	0.9065	0.8945	0.9169	0.9045	0.9582	**0.8839**
	0.71	16.15	0.03	6.24	2.94	18.17
0.3	0.9395	0.9127	0.9336	0.9218	0.9716	**0.8850**
	0.72	18.12	0.03	6.24	3.02	19.92
0.4	1.0083	0.9462	0.9553	0.9433	0.9652	**0.9133**
	0.71	12.12	0.03	6.31	3.12	15.49
0.5	1.0246	0.9649	0.9733	0.9610	0.9989	**0.9371**
	0.72	13.93	0.03	6.71	3.25	19.43
0.6	1.0338	0.9759	0.9861	0.9756	1.0184	**0.9609**
	0.69	16.07	0.03	6.76	3.26	19.51
0.7	1.0510	0.9930	0.9940	0.9858	1.0264	**0.9752**
	0.69	18.59	0.03	6.96	3.33	15.36
0.8	1.0581	1.0018	0.9998	0.9922	1.0358	**0.9921**
	0.69	16.35	0.03	6.99	3.43	18.38
0.9	1.0484	1.0089	0.9995	**0.9964**	1.0507	1.0059
	0.69	16.76	0.03	6.92	3.49	17.42
1.0	1.0478	1.0212	**1.0000**	**1.0000**	1.0254	1.0048
	0.35	5.21	0.02	3.47	1.95	4.56
Best	0.85	0.84	0.85	0.84	0.84	0.79

Table 8. Results for the 8x8 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.8396	**0.8387**	0.8892	0.8581	0.9624	0.8637
	1.50	47.98	0.05	35.00	6.37	69.63
0.2	0.8620	0.8539	0.8809	**0.8491**	0.9534	0.8664
	1.49	38.75	0.05	33.56	6.01	42.08
0.3	0.9646	0.8956	0.9137	**0.8780**	0.9743	0.8832
	1.51	38.91	0.04	35.44	7.02	54.01
0.4	0.9957	0.9371	0.9476	**0.9096**	1.0194	0.9352
	1.47	35.34	0.04	44.31	7.13	53.33
0.5	1.0199	0.9560	0.9732	**0.9393**	1.0637	0.9628
	1.51	32.68	0.04	47.02	7.55	57.29
0.6	1.0393	0.9711	0.9901	**0.9606**	1.0688	0.9787
	1.46	54.79	0.06	43.07	7.48	55.16
0.7	1.0440	0.9824	0.9995	**0.9766**	1.0725	0.9816
	1.46	39.21	0.04	57.72	7.67	46.09
0.8	1.0674	0.9959	1.0025	**0.9870**	1.1342	1.0370
	1.46	48.79	0.05	58.07	7.84	47.35
0.9	1.0637	0.9979	1.0019	**0.9940**	1.1345	1.0224
	1.46	49.89	0.05	38.67	8.05	62.26
1.0	1.0675	**1.0000**	**1.0000**	**1.0000**	1.0945	1.0105
	0.72	14.02	0.03	7.54	5.01	30.73
Best	0.79	0.79	0.83	0.77	0.86	0.75

Table 9. Results for the 9x9 grid network

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9954 0.00	0.9847 0.01	1.0012 0.00	0.9863 0.00	0.9924 0.00	**0.9845** 0.01
0.2	1.0062 0.00	0.9879 0.01	1.0078 0.00	0.9891 0.01	1.0055 0.00	**0.9878** 0.01
0.3	1.0140 0.00	0.9872 0.01	1.0129 0.00	0.9882 0.01	1.0054 0.00	**0.9851** 0.00
0.4	1.0119 0.00	0.9862 0.00	1.0159 0.00	0.9874 0.00	1.0053 0.00	**0.9846** 0.00
0.5	1.0113 0.00	**0.9839** 0.01	1.0169 0.00	0.9873 0.01	1.0002 0.00	0.9873 0.01
0.6	1.0106 0.00	0.9884 0.01	1.0160 0.00	**0.9881** 0.00	1.0044 0.00	0.9903 0.00
0.7	1.0318 0.00	**0.9881** 0.01	1.0134 0.00	0.9899 0.01	1.0050 0.00	0.9900 0.01
0.8	1.0200 0.00	**0.9912** 0.01	1.0095 0.00	0.9926 0.01	1.0006 0.00	**0.9912** 0.01
0.9	1.0145 0.00	**0.9953** 0.01	1.0049 0.00	0.9960 0.01	1.0003 0.00	**0.9953** 0.00
1.0	1.0509 0.00	**1.0000** 0.00	**1.0000** 0.00	**1.0000** 0.00	**1.0000** 0.00	**1.0000** 0.00
Best	0.98	0.96	0.96	0.96	0.96	0.96

Table 10. Results for 8 node Euclidean network (density – 0.3)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9947 0.00	0.9836 0.04	0.9976 0.00	0.9828 0.04	1.0001 0.00	**0.9824** 0.03
0.2	1.0000 0.00	0.9746 0.04	1.0005 0.00	**0.9734** 0.04	0.9952 0.01	0.9765 0.03
0.3	1.0053 0.00	0.9734 0.03	1.0015 0.00	**0.9693** 0.04	1.0035 0.01	0.9745 0.04
0.4	1.0200 0.00	0.9736 0.05	1.0020 0.00	**0.9718** 0.04	1.0019 0.01	0.9737 0.04
0.5	1.0139 0.00	0.9713 0.04	1.0022 0.00	0.9733 0.04	1.0116 0.00	**0.9706** 0.03
0.6	1.0143 0.00	0.9749 0.04	1.0022 0.00	0.9750 0.03	1.0222 0.01	**0.9747** 0.03
0.7	1.0155 0.00	0.9819 0.05	1.0018 0.00	**0.9808** 0.03	1.0218 0.00	0.9827 0.03
0.8	1.0125 0.00	**0.9851** 0.05	1.0013 0.00	0.9879 0.03	1.0241 0.01	0.9853 0.03
0.9	1.0145 0.00	0.9948 0.04	1.0006 0.00	0.9930 0.03	1.0180 0.01	**0.9910** 0.04
1.0	1.0068 0.00	**1.0000** 0.02	**1.0000** 0.00	**1.0000** 0.02	1.0313 0.00	**1.0000** 0.02
Best	0.95	0.93	0.97	0.94	0.96	0.93

Table 11. Results for 8 node Euclidean network (density – 0.5)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9952 0.00	0.9873 0.09	1.0035 0.00	**0.9852** 0.09	0.9969 0.02	0.9872 0.06
0.2	1.0031 0.00	0.9795 0.11	1.0040 0.00	**0.9755** 0.10	0.9921 0.02	0.9763 0.09
0.3	1.0030 0.00	0.9747 0.08	1.0039 0.00	0.9760 0.09	0.9899 0.02	**0.9708** 0.08
0.4	1.0001 0.01	**0.9722** 0.10	1.0029 0.00	0.9770 0.08	0.9909 0.02	0.9768 0.08
0.5	1.0026 0.00	**0.9767** 0.10	1.0018 0.00	0.9801 0.07	0.9940 0.02	0.9788 0.06
0.6	1.0085 0.00	0.9813 0.08	1.0010 0.00	0.9819 0.07	0.9977 0.02	**0.9789** 0.07
0.7	1.0124 0.00	0.9845 0.08	1.0008 0.00	0.9829 0.08	0.9912 0.03	**0.9810** 0.08
0.8	1.0071 0.01	0.9878 0.07	1.0008 0.00	**0.9872** 0.08	0.9967 0.02	0.9877 0.07
0.9	1.0042 0.00	0.9938 0.08	1.0008 0.00	0.9936 0.07	0.9994 0.03	**0.9932** 0.08
1.0	1.0007 0.00	**1.0000** 0.04	**1.0000** 0.00	**1.0000** 0.04	**1.0000** 0.02	**1.0000** 0.04
Best	0.95	0.93	0.98	0.93	0.95	0.93

Table 12. Results for 8 node Euclidean network (density – 0.7)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9958 0.00	**0.9822** 0.04	0.9950 0.00	0.9857 0.03	0.9984 0.00	0.9852 0.04
0.2	1.0036 0.00	0.9809 0.03	1.0000 0.00	0.9756 0.03	0.9898 0.00	**0.9746** 0.03
0.3	1.0210 0.00	0.9754 0.03	1.0040 0.00	0.9696 0.04	0.9918 0.00	**0.9677** 0.04
0.4	1.0184 0.00	0.9711 0.03	1.0065 0.00	**0.9686** 0.03	1.0111 0.00	0.9687 0.03
0.5	1.0315 0.00	0.9648 0.04	1.0078 0.00	**0.9608** 0.03	1.0314 0.00	0.9685 0.03
0.6	1.0207 0.00	**0.9620** 0.04	1.0080 0.00	0.9637 0.03	1.0217 0.00	0.9707 0.04
0.7	1.0235 0.00	0.9728 0.03	1.0071 0.00	**0.9679** 0.03	1.0209 0.00	0.9709 0.04
0.8	1.0203 0.00	**0.9790** 0.03	1.0054 0.00	0.9818 0.03	1.0229 0.00	0.9802 0.03
0.9	1.0226 0.00	0.9914 0.03	1.0029 0.00	0.9901 0.03	1.0234 0.00	**0.9889** 0.03
1.0	1.0149 0.00	1.0011 0.02	**1.0000** 0.00	**1.0000** 0.02	1.0189 0.00	1.0011 0.02
Best	0.94	0.91	0.97	0.91	0.95	0.92

Table 13. Results for 10 node Euclidean network (density – 0.3)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0009 0.01	0.9866 0.20	1.0004 0.00	0.9848 0.21	0.9980 0.03	**0.9842** 0.14
0.2	1.0110 0.00	0.9798 0.18	1.0015 0.00	0.9747 0.15	0.9974 0.03	**0.9733** 0.17
0.3	1.0073 0.00	0.9753 0.17	1.0027 0.00	**0.9687** 0.19	1.0085 0.03	0.9758 0.14
0.4	1.0069 0.00	0.9753 0.14	1.0035 0.00	0.9786 0.18	1.0057 0.03	**0.9746** 0.19
0.5	1.0052 0.00	0.9790 0.15	1.0038 0.00	0.9802 0.16	1.0076 0.04	**0.9788** 0.15
0.6	1.0089 0.00	0.9793 0.15	1.0034 0.00	0.9791 0.17	1.0098 0.04	**0.9786** 0.14
0.7	1.0115 0.01	0.9833 0.15	1.0026 0.00	**0.9812** 0.18	1.0124 0.04	**0.9812** 0.16
0.8	1.0113 0.00	0.9873 0.17	1.0016 0.00	0.9868 0.15	1.0133 0.04	**0.9841** 0.18
0.9	1.0170 0.01	0.9924 0.17	1.0007 0.00	0.9922 0.14	1.0178 0.04	**0.9911** 0.17
1.0	1.0064 0.00	**1.0000** 0.07	**1.0000** 0.00	**1.0000** 0.07	1.0157 0.03	1.0008 0.08
Best	0.98	0.95	0.98	0.94	0.97	0.95

Table 14. Results for 10 node Euclidean network (density – 0.5)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0038 0.02	0.9879 0.42	1.0018 0.00	0.9861 0.39	1.0038 0.09	**0.9856** 0.44
0.2	1.0077 0.01	0.9853 0.36	1.0050 0.00	**0.9794** 0.44	1.0068 0.09	0.9818 0.44
0.3	1.0079 0.01	0.9836 0.39	1.0068 0.00	0.9792 0.51	1.0067 0.09	**0.9790** 0.43
0.4	1.0136 0.01	0.9863 0.39	1.0070 0.00	0.9844 0.40	1.0074 0.09	**0.9842** 0.35
0.5	1.0148 0.01	0.9910 0.45	1.0062 0.00	0.9875 0.37	1.0064 0.10	**0.9845** 0.33
0.6	1.0165 0.01	0.9905 0.40	1.0049 0.00	0.9903 0.32	1.0101 0.10	**0.9853** 0.44
0.7	1.0097 0.01	0.9907 0.31	1.0035 0.00	0.9910 0.33	1.0120 0.10	**0.9897** 0.46
0.8	1.0082 0.01	0.9924 0.30	1.0022 0.00	**0.9920** 0.43	1.0143 0.11	0.9923 0.29
0.9	1.0094 0.02	0.9966 0.32	1.0011 0.00	**0.9947** 0.39	1.0210 0.11	0.9964 0.36
1.0	1.0086 0.01	1.0017 0.16	**1.0000** 0.00	**1.0000** 0.14	1.0087 0.07	1.0033 0.15
Best	0.98	0.96	0.99	0.96	0.98	0.97

Table 15. Results for 10 node Euclidean network (density – 0.7)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0076 0.01	0.9802 0.39	0.9921 0.00	**0.9757** 0.36	0.9976 0.08	0.9812 0.37
0.2	0.9996 0.02	0.9611 0.51	0.9884 0.00	**0.9597** 0.34	0.9999 0.08	0.9648 0.31
0.3	1.0159 0.01	0.9648 0.37	0.9877 0.00	**0.9588** 0.38	1.0037 0.09	0.9634 0.45
0.4	1.0041 0.02	0.9676 0.41	0.9882 0.00	**0.9607** 0.38	1.0057 0.09	0.9616 0.36
0.5	1.0028 0.02	0.9758 0.25	0.9893 0.00	**0.9646** 0.35	1.0094 0.09	0.9700 0.31
0.6	1.0122 0.01	0.9730 0.35	0.9907 0.00	**0.9688** 0.38	1.0196 0.09	0.9772 0.32
0.7	1.0158 0.02	0.9752 0.39	0.9926 0.00	**0.9728** 0.35	1.0145 0.10	0.9787 0.37
0.8	1.0149 0.01	0.9848 0.40	0.9948 0.00	**0.9816** 0.31	1.0109 0.10	0.9858 0.33
0.9	1.0253 0.01	0.9913 0.33	0.9973 0.00	**0.9908** 0.27	1.0086 0.10	0.9930 0.29
1.0	1.0222 0.01	1.0024 0.17	**1.0000** 0.00	**1.0000** 0.14	1.0324 0.06	1.0031 0.17
Best	0.96	0.93	0.95	0.92	0.93	0.92

Table 16. Results for 15 node Euclidean network (density – 0.3)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0058 0.07	0.9794 2.20	0.9996 0.01	**0.9777** 1.85	1.0023 0.40	0.9807 1.66
0.2	1.0047 0.08	0.9724 2.04	0.9983 0.00	0.9726 1.96	1.0013 0.41	**0.9690** 1.94
0.3	1.0057 0.07	0.9731 1.96	0.9968 0.01	**0.9728** 2.15	0.9997 0.42	0.9798 1.86
0.4	1.0050 0.08	0.9768 1.73	0.9959 0.00	0.9792 1.73	0.9968 0.44	**0.9762** 1.80
0.5	1.0023 0.07	**0.9789** 1.92	0.9955 0.01	0.9805 1.72	0.9995 0.45	0.9831 1.58
0.6	1.0068 0.07	0.9851 1.78	0.9959 0.01	**0.9834** 1.74	1.0077 0.47	0.9875 1.52
0.7	1.0045 0.07	0.9862 1.77	0.9969 0.01	**0.9855** 1.77	1.0068 0.48	0.9877 1.89
0.8	1.0058 0.07	0.9926 1.23	0.9983 0.00	**0.9908** 1.35	1.0149 0.49	0.9925 1.51
0.9	1.0098 0.07	0.9953 1.78	0.9995 0.01	**0.9949** 1.23	1.0209 0.51	0.9956 1.74
1.0	1.0092 0.03	**1.0000** 0.77	**1.0000** 0.00	**1.0000** 0.56	1.0234 0.31	1.0003 0.88
Best	0.99	0.95	0.98	0.96	0.96	0.96

Table 17. Results for 15 node Euclidean network (density – 0.5)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0034 0.19	0.9851 5.28	0.9966 0.02	**0.9808** 3.00	0.9996 1.08	0.9838 4.86
0.2	1.0028 0.19	0.9814 4.90	0.9929 0.02	**0.9755** 4.47	0.9998 1.09	0.9807 5.16
0.3	0.9999 0.20	0.9819 4.66	0.9897 0.01	**0.9762** 4.04	0.9976 1.13	0.9785 5.85
0.4	1.0020 0.19	0.9827 5.00	0.9882 0.03	**0.9750** 4.36	0.9967 1.20	0.9831 3.90
0.5	0.9998 0.20	0.9826 4.45	0.9883 0.02	**0.9806** 3.29	1.0001 1.23	0.9842 5.58
0.6	1.0019 0.19	0.9874 4.99	0.9900 0.02	**0.9831** 3.19	1.0044 1.29	0.9867 5.12
0.7	1.0031 0.19	0.9918 3.77	0.9926 0.02	**0.9886** 2.64	1.0026 1.31	0.9911 4.15
0.8	1.0032 0.20	0.9938 3.97	0.9958 0.02	**0.9924** 2.93	1.0087 1.38	0.9956 3.57
0.9	1.0031 0.18	0.9967 3.34	0.9986 0.03	**0.9964** 3.14	1.0097 1.41	0.9969 3.68
1.0	1.0052 0.10	**1.0000** 1.64	**1.0000** 0.02	**1.0000** 1.31	1.0128 0.82	**1.0000** 1.75
Best	0.98	0.96	0.97	0.95	0.98	0.97

Table 18. Results for 15 node Euclidean network (density – 0.7)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	0.9959 0.09	**0.9676** 2.30	0.9991 0.00	0.9691 2.04	0.9934 0.63	0.9699 3.08
0.2	0.9978 0.09	0.9664 2.67	0.9964 0.01	**0.9603** 2.29	0.9963 0.66	0.9635 2.64
0.3	1.0032 0.10	0.9635 3.09	0.9931 0.01	0.9619 2.37	0.9973 0.68	**0.9588** 2.91
0.4	1.0069 0.09	0.9766 2.52	0.9910 0.01	**0.9664** 2.82	0.9984 0.69	0.9734 2.48
0.5	1.0079 0.10	0.9724 2.04	0.9905 0.00	**0.9703** 2.46	1.0014 0.82	0.9717 2.61
0.6	1.0114 0.08	0.9788 2.20	0.9914 0.01	**0.9733** 2.31	1.0071 0.78	0.9782 2.13
0.7	1.0083 0.09	0.9817 2.00	0.9932 0.01	**0.9793** 1.99	1.0068 0.81	0.9828 2.38
0.8	1.0127 0.10	0.9867 2.49	0.9956 0.00	**0.9851** 1.76	1.0185 0.84	0.9881 2.43
0.9	1.0161 0.10	0.9922 2.83	0.9980 0.01	**0.9917** 1.80	1.0166 0.86	0.9946 2.35
1.0	1.0166 0.05	**1.0000** 1.06	**1.0000** 0.00	**1.0000** 0.72	1.0307 0.39	1.0006 1.12
Best	0.97	0.94	0.98	0.94	0.97	0.92

Table 19. Results for 20 node Euclidean network (density – 0.3)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0001 0.40	0.9788 11.02	1.0000 0.04	**0.9746** 14.16	0.9993 2.90	0.9796 13.55
0.2	1.0013 0.41	0.9770 12.61	0.9980 0.03	0.9757 12.53	0.9962 2.99	**0.9756** 11.26
0.3	1.0036 0.43	0.9762 14.81	0.9960 0.03	**0.9739** 16.43	1.0022 3.14	0.9749 14.34
0.4	1.0001 0.40	0.9796 11.23	0.9948 0.03	**0.9769** 14.63	0.9967 3.39	0.9783 14.97
0.5	1.0029 0.41	0.9833 11.20	0.9948 0.04	**0.9818** 10.19	1.0020 3.86	0.9847 12.31
0.6	1.0038 0.40	0.9874 12.46	0.9956 0.03	**0.9866** 9.18	1.0036 3.66	0.9870 12.55
0.7	1.0049 0.41	0.9914 10.07	0.9970 0.04	**0.9881** 9.85	1.0091 3.83	0.9911 11.95
0.8	1.0047 0.41	0.9937 11.58	0.9986 0.03	**0.9919** 11.92	1.0095 3.93	0.9958 12.26
0.9	1.0069 0.41	0.9963 13.91	0.9999 0.04	**0.9960** 11.83	1.0104 4.02	0.9986 11.15
1.0	1.0061 0.21	1.0006 4.32	**1.0000** 0.02	**1.0000** 3.09	1.0176 1.76	1.0021 3.73
Best	0.99	0.96	0.99	0.96	0.98	0.96

Table 20. Results for 20 node Euclidean network (density – 0.5)

Prob.	Heur 1	H1+DA	Heur 2	H2+DA	Heur 3	H3+DA
0.1	1.0008	0.9813	0.9917	**0.9684**	0.9923	0.9764
	1.12	32.75	0.08	38.08	6.30	34.27
0.2	1.0000	0.9761	0.9845	**0.9665**	0.9947	0.9720
	1.12	35.44	0.08	29.97	6.31	43.79
0.3	0.9936	0.9718	0.9810	**0.9676**	0.9962	0.9719
	1.14	47.44	0.08	35.85	6.54	49.35
0.4	0.9962	0.9771	0.9807	**0.9697**	0.9963	0.9782
	1.12	36.66	0.08	32.38	6.68	33.26
0.5	0.9969	0.9806	0.9826	**0.9758**	0.9953	0.9811
	1.13	31.46	0.08	24.82	6.97	35.93
0.6	0.9985	0.9877	0.9861	**0.9804**	0.9987	0.9854
	1.12	26.96	0.08	29.14	7.66	34.41
0.7	0.9996	0.9899	0.9906	**0.9867**	1.0054	0.9906
	1.11	37.92	0.08	25.19	8.05	36.07
0.8	1.0013	0.9941	0.9951	**0.9920**	1.0060	0.9951
	1.12	25.96	0.08	29.53	8.46	27.70
0.9	1.0032	0.9978	0.9988	**0.9969**	1.0126	0.9977
	1.12	28.37	0.08	30.65	9.14	30.58
1.0	1.0024	**1.0000**	**1.0000**	**1.0000**	1.0170	1.0004
	0.56	9.89	0.04	7.64	4.80	10.93
Best	0.98	0.96	0.97	0.95	0.98	0.96

Table 21. Results for 20 node Euclidean network (density – 0.7)

Prob.	Grid	Euclidean		
		0.3	0.5	0.7
0.1	0.9013	0.9775	0.9797	0.9800
0.2	0.9005	0.9706	0.9728	0.9742
0.3	0.8984	0.9676	0.9712	0.9734
0.4	0.9205	0.9701	0.9749	0.9753
0.5	0.9430	0.9699	0.9775	0.9794
0.6	0.9577	0.9731	0.9808	0.9819
0.7	0.9723	0.9770	0.9839	0.9865
0.8	0.9819	0.9842	0.9880	0.9909
0.9	0.9913	0.9917	0.9933	0.9953
1.0	0.9986	1.0000	1.0000	1.0000

Table 22. Overall average of best results for grid and Euclidean networks

4.4.2 PERFORMANCE OF THE HEURISTICS

Our results indicate that when the probability of occurrence of the white edges is very low ($p = 0.1$), the global greedy heuristic along with DROP-ADD seems to perform better than the other two heuristics. For other values of p, the sub-tour construction heuristic along with DROP-ADD clearly seems to perform better than the other two heuristics for grid networks.

For the Euclidean networks, as the number of edges increases, the local greedy heuristic along with DROP-ADD seems to perform the best among the three, though the margin of improvement is very small. For problems with smaller number of white edges, the best results seem to be spread among the three heuristics for the various values of p.

The results on our computational times indicate that all the three heuristics are quite fast. For the largest problems (9x9 grid networks, and 20 node Euclidean networks with edge density of 0.7), all the heuristics (including the post-optimization phase) generate tours in about a minute. So we can generate tours using all the three heuristics and choose the tour with the best expected length.

4.4.3 COMPARISON OF HEURISTICS WITH RANDOM TOUR

We also compared the expected lengths of the tours produced by the three heuristics with the expected length of a random Eulerian tour. Table 22 contains average results for the grid and Euclidean networks. For a given probability, we pick the results produced by the heuristic with the best average result over 10 instances for each problem size, and compute the overall average best result over all the problem sizes using these results. Our results show that for the grid networks, the expected length of our overall average best solution is lower than the expected length of a random tour by 10% on average, for lower values of p. As p increases to 1.0, this average reduces to 6% for $p = 0.5$ and 1% for $p = 0.9$. In one particular 9x9 instance, the expected length of the tour generated by the sub-tour construction heuristic is 25% lower than the expected length of the random Eulerian tour. For the Euclidean

networks, the gaps are not as dramatic. The expected length of our overall average best solution is lower than the expected length of a random tour by 2% on average, for low values of p. These results clearly show that it is advantageous to use heuristics designed specifically for the SETP rather than generate a random Eulerian tour.

4.5 CONCLUSION

In this chapter, we have presented three different heuristics for the SETP and the DROP-ADD post-optimization procedure. The global greedy heuristic starts with an empty tour and determines the next edge to service as the one that results in the minimum expected increase in the length when appended at the end of the tour. The local greedy heuristic selects the next edge of the tour from the set of edges at the current node rather than from the set of all available white edges. Finally the sub-tour construction heuristic constructs several small sub-tours and then concatenates them while considering the expected savings in concatenating sub-tours.

We have tested the performance of the three heuristics on grid networks and on Euclidean graphs of various sizes. Our results indicate that the sub-tour construction heuristic performs well when $p > 0.1$ for the grid networks. For the Euclidean networks, as the number of edges increases, the second heuristic performs the best among the three, though the margin of improvement is small.

We also compared the expected lengths of the tours produced by the three heuristics with the expected length of a random Eulerian tour. Our results show that for the grid networks, the expected length of our overall average best solution is lower than the expected length of a random tour by 10% on average, for lower values of p. As p increases to 1.0, this average reduces to 6% for $p = 0.5$ and 1% for $p = 0.9$. For the Euclidean networks, the expected length of our overall average best solution is lower than the expected length of a random tour by 2% on average, for low values of p. This chapter is a good start to the methodological contribution to the SETP and the results can definitely be used as a starting point for the development of meta-heuristics for the SETP.

CHAPTER 5

CONCLUSION

5.1 CONTRIBUTIONS OF THIS THESIS

This thesis focused on two specific and important problems in arc routing – the *Capacitated Arc Routing Problem* (CARP) and the *Stochastic Eulerian Tour Problem* (SETP). Both problems have excellent application potential and deserve special attention.

5.1.1 THEORETICAL CONTRIBUTION

The theoretical contribution of this thesis is through the *Stochastic Eulerian Tour Problem* (SETP), the second problem that we considered in this thesis. As explained in the previous chapters, the SETP arises when the set of edges that have to be visited on any particular day is random. The investigation of this problem was actually motivated by the existence of a real world problem. In the UK postal system, the carriers deliver mail a second time in the afternoon when the number of streets to be visited is very small and varies from day to day. Given this scenario, the mail carrier, while following his regular route, usually skips the streets that do not require a visit. Thus, given an undirected graph $G = (V, E)$ where all the edges in E require service, a distance matrix D and a probability distribution for the number of required edges present, the SETP seeks an a priori Eulerian tour of minimum expected length.

This thesis has defined and investigated this problem for the first time. We feel that it plays an important role in scenarios where the number of edges to be visited each day is random and smaller compared to the total number of edges that require service.

We have derived a closed form expression for the expected length of a given tour when the number of present edges follows a binomial distribution. This result can be easily extended to the situation where the number of white edges present follows any discrete probability distribution if the set of present white edges can be chosen randomly from the set of all white edges. We have also shown that the SETP is NP-hard, even though the deterministic counter part is solvable in polynomial time. We have derived further properties and a worst case ratio for the deviation of the expected length of a random Eulerian tour from the optimal tour in the expected sense.

5.1.2 METHODOLOGICAL CONTRIBUTIONS

The methodological contributions of this thesis are the new tabu search algorithm for the CARP that considers work load balancing, and the three heuristics for the SETP. The CARP is one of the most important problems in arc routing due to its presence in applications such as snow plowing, street cleaning, garbage collection, mail delivery, and many others. The CARP is a very hard problem, and it is quite unrealistic to believe that exact procedures can be used to solve even average sized problems. Researchers have developed several heuristics for the CARP. Most of these heuristics are simple one-shot heuristics. Recently, Hertz, Laporte, and Mittaz (1996) have developed a tabu search based heuristic, CARPET, for the CARP that incorporates local improvement routines.

We have developed a new tabu search based algorithm, TABUCARP, that considers a secondary objective of balancing the total work load (demand) fairly equally among the routes, in addition to minimizing the total cost. We feel that it is quite important to incorporate this feature, since most applications such as mail delivery, meter reading, and garbage collection require work load balancing. In a real-world scenario, if the existing algorithm considers only the total cost, generally, the planner revisits the solution and moves demand among the various routes myopically in order to balance the load. On the other hand, our procedure (TABUCARP) aims to be a little more global and builds this additional feature into the algorithm. This is an important contribution of this thesis since this is a first step towards using TABUCARP as a useful planning tool for several arc routing applications.

We have tested TABUCARP on a set of 23 test problems by DeArmon, and another set of random problems. TABUCARP produces routes similar to CARPET for the DeArmon problems. On the random problems, the total distance traversed by TABUCARP's solutions is 2.78% higher than CARPET's solution on average. However, for this relatively small increase we get better work load balancing. The coefficient of variation for work load balance is 7.69% lower for TABUCARP's solutions.

The SETP belongs to a class of hard problems, and hence, it is not possible to solve realistic sized problems optimally using algorithms that would run in polynomial time. Hence, we have developed three heuristics for the SETP. The first heuristic, a global greedy heuristic, starts with an empty tour and determines the next edge to service as the one that results in the minimum expected increase in the length when appended at the end of the tour. The second heuristic, a local greedy heuristic, selects the next edge of the tour from the set of edges at the current node rather than from the set of all available white edges. Finally the third heuristic, a sub-tour construction heuristic, constructs several small sub-tours and then concatenates these sub-tours while considering the expected savings in concatenating sub-tours. We have also incorporated an adaptation of the US post-optimization procedure developed by Gendreau et al. (1992) for the TSP.

We have tested the performance of the three heuristics on grid networks and Euclidean graphs of various sizes. Our results indicate that when the probability of occurrence of the white edges is very low ($p = 0.1$), the global greedy heuristic seems to perform better the other two heuristics. In other situations, the sub-tour construction heuristic seems to perform well for the grid networks. For the Euclidean networks, as the number of edges increases, the second heuristic seems to perform the best among the three, though the margin of improvement is small.

We also compared the expected lengths of the tours produced by the three heuristics with the expected length of a random Eulerian tour. Our results show that for the grid networks, the expected length of our best solution is lower than the expected length of a random tour by 10% on average, for lower values of p. For the Euclidean networks, the expected length of our best solution is lower than the expected length of a random tour by 2% on average, for low values of p.

5.2 DIRECTIONS FOR FUTURE RESEARCH

This thesis has addressed two important problems in the area of arc routing. There are several theoretical and practical issues that we would like to investigate further as part of continuing research in this area.

- The TABUCARP algorithm that we have developed seems to perform well in terms of producing fairly balanced routes. As a first step towards developing a planning tool for several applications, we would like to test and validate the algorithm with real world data. Once this process is completed successfully, TABUCARP could form the backbone for developing a user friendly scheduling tool for several arc routing applications.
- The definition and formulation of the SETP has made it possible to investigate the use of the SETP methodology in actual applications such as the UK postal situation. The results from Chapter 4 indicate that even a simple application of the concepts developed in this thesis could result in a reasonable reduction of the expected lengths of the postman tours in these applications. Here again, the use of real world data would help validate the methodology.
- We would like to explore the possibility of developing exact solution procedures for the SETP. One possibility is to formulate the problem as a stochastic integer program, and use the integer L-shaped method to devise a branch-and-cut algorithm for the SETP. Obtaining the optimal solutions for possible instances would also help assess the accuracy of the heuristics we have developed in Chapter 4.
- The sub-tour construction heuristic presented in Chapter 4 does not fully exploit the desirable properties of a good a priori tour. We can develop meta-heuristics such as tabu search based heuristics that moves edges among sub-tours to produce a large number of small and balanced sub-tours. The concept of work load balancing from TABUCARP can be extended to this situation to balance the size of the various sub-tours.
- Finally, for the SETP, this thesis focussed on finding the Eulerian tour of minimum expected length after the given graph is made Eulerian by solving the minimum cost augmentation problem. However, based on Jaillet's results for the PTSP, we feel that

it will be worth the effort to consider the augmentation problem and the SETP simultaneously. In this situation, it might be possible to develop an Eulerian tour of smaller expected length compared to a random Eulerian tour on an augmented graph, even when all the white edges are present in the given graph.

We hope that this thesis has provided the motivation for exploring the SETP and several other stochastic arc routing problems further.

REFERENCES

Assad, A. A. and B. L. Golden, "Arc Routing Methods and Studies," in *Network Routing*, Handbooks in Operations Research and Management Science, M. O. Ball, T. L. Magnanti, C. L. Monma and G. L. Nemhauser (eds.), North-Holland, Amsterdam, 1995.

Assad, A. A., W. -L. Pearn, B. L. Golden, "The Capacitated Chinese Postman Problem: Lower Bounds and Solvable Cases," *American Journal of Mathematical and Management Sciences*, 7 (1987), 63-88.

Barahona, F., "On Some Applications of the Chinese Postman Problem," in *Paths, Flows and VLSI-Layout*, Algorithms and Combinatorics, 9, B. Korte, L. Lovász, H. J. Promel, and A. Schriver, Eds., Springer-Verlag, Berlin, 1990.

Bartholdi, J. J. and L. K. Platzman, "An (N log N) Planar Traveling Salesman Heuristic Based on Spacefilling Curves," *Operations Research Letters*, 1 (1982), 121-125.

Belenguer, J. M. and E. Benavent, "Polyhedral Results on the Capacitated Arc Routing Problem," Working Paper, Departmento de Estadistica e Investigacion Operativa, Universidad de Valencia (1991).

Beltrami, E. L. and L.D. Bodin, "Networks and Vehicle Routing for Municipal Waste Collection," *Networks*, 4 (1974), 65-94.

Benavent, E., V. Campos, A. Corberán, and E. Mota, "The Capacitated Arc Routing Problem: A Heuristic Algorithm," *Quaderns d'Estadistica, Sistemes, Informatica I Investigacio Operativa (QUESTIIO)*, 14 (1990), 107-122.

Benavent, E., V. Campos, A. Corberán, and E. Mota, "The Capacitated Arc Routing Problem: Lower Bounds," *Networks*, 22 (1992), 669-690.

Bennet, B. and D. Gazis, "School Bus Routing by Computer," *Transportation Research*, 6 (1972), 317-326.

Bertsimas, D. J., Probabilistic Combinatorial Optimization Problems, Ph. D. Thesis, Report No. 193, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA, 1988.

Bertsimas, D. J. and L. H. Howell, "Further Results on the Probabilistic Traveling Salesman Problem," *European Journal of Operational Research*, 65 (1993), 68-95.

Bodin, L. D. and O. Berman, "Routing and Scheduling of School Buses by Computer," *Transportation Science*, 13 (1979), 113-129.

Bodin, L. D., G. Fagin, R. Welebny, and J. Greenberg, "The Design of a Computerized Sanitation Vehicle Routing and Scheduling system for the Town of Oyster Bay, New York," *Computers & Operations Research*, 16 (1989), 45-54.

Bodin, L. D., B. L. Golden, A. A. Assad, and M. O. Ball, "Routing and Scheduling of Vehicles and Crews. The State of the Art," *Computers & Operations Research*, 10 (1983), 63-211.

Bodin, L. D. and S. J. Krush, "A Computer-Assisted System for the Routing and Scheduling of Street Sweepers," *Operations Research*, 26 (1978), 525-537.

Bouliane, J. and G. Laporte, "Locating Postal Relay Boxes Using a Set Covering Algorithm," *American Journal of Mathematical and Management Sciences*, 12 (1992), 65-74.

Brandimarte, P., "Routing and Scheduling in a Flexible Job Shop by Tabu Search," *Annals of Operations Research*, 41 (1993), 327-342.

Brucker, P., "The Chinese Postman Problem for Mixed Graphs," in *Graph Theoretic Concepts in Computer Science*, H. Noltemeier (ed.), Springer-Verlag, Berlin, 1981.

Busacker, R. G. and T. L. Saaty, *Finite Graphs and Networks*, McGraw-Hill, New York, 1975.

Chapleau, L., J. A. Ferland, G. Lapalme, and J.-M. Rousseau, "A Parallel-Insert Method for the Capacitated Arc Routing Problem," *Operations Research Letters*, 3 (1984), 95-99.

Christofides, N., "The Optimum Traversal of a Graph," *Omega*, 1 (1973), 719-732.

Christofides, N., E. Benavent, V. Campos, A. Corberán, and E. Mota, "An Optimal Method for the Mixed Postman Problem," in *System Modelling and Optimization*, Lecture Notes in Control and information Sciences, 59, P. Thoft-Christensen (ed.), Springer-Verlag, Berlin, 1984.

Christofides, N., V. Campos, A. Corberán, and E. Mota, "An Algorithm for the Rural Postman Problem," Imperial College Report IC.O.R. 81.5, 1981.

Christofides, N., V. Campos, A. Corberán, and E. Mota, "An Algorithm for the Rural Postman Problem on a Directed Graph," *Mathematical Programming Study*, 26 (1986), 155-166.

Clark, R. M. and J. I. Gillean, "Analysis of solid Waste Management Operations in Cleveland, Ohio," *Interfaces*, 6 (1975), 32-42.

Clark, R. M. and C. H. Lee, Jr., "Systems Planning for Solid Waste Collection," *Computers & Operations Research*, 3 (1976), 157-173.

Clarke, G. and J. M. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," *Operations Research*, 12 (1964), 568-581.

Cook, T. M. and B. S. Alprin, "Snow and Ice Removal in an Urban Environment," *Management Science*, 23 (1976), 227-234.

Daniels, R. L. and J. B. Mazola, "A Tabu Search Heuristic for the Flexible-Resource Flow Shop Scheduling Problem," *Annals of Operations Research*, 41 (1993), 207-230.

Dell'Amico, M. and T. Trubian, "Applying Tabu Search to the Job-Shop Scheduling Problem," *Annals of Operations Research*, 41 (1993), 231-252.

Desrosiers, J., J. A. Ferland, J. -M. Rousseau, G. Lapalme, and L. Chapleau, "TRANSCOL: A Multi-Period School Bus Routing and Scheduling System," *TIMS Studies in the Management Sciences*, 22 (1986), 47-71.

Dror, M., H. Stern, and P. Trudeau, "Postman Tour on a graph with Precedence Relation on Arcs," *Networks*, 17 (1987), 283-294.

Edmonds, J., "The Chinese Postman's Problem," *ORSA Bulletin*, 13 (1965a), 73.

Edmonds, J., "Paths, Trees, and Flowers," *Canadian Journal of Mathematics*, 17 (1965b), 449-467.

Edmonds, J. and E. L. Johnson, "Matching, Euler Tours and the Chinese Postman Problem," *Mathematical Programming*, 5 (1973), 88-124.

Eglese, R. W. and L. Y. O. Li, "Efficient Routeing for Winter Gritting," *Journal of the Operational Research Society*, 43 (1992), 1031-1034.

Eglese, R. W. and H. Murdock, "Routeing Road Sweepers in a Rural Area," *Journal of the Operational Research Society*, 42 (1991), 281-288.

Eiselt, H. A., M. Gendreau, and G. Laporte, "Arc Routing Problems. Part I: The Chinese Postman Problem," *Operations Research*, 43 (1995), 231-242.

Eiselt, H. A., M. Gendreau, and G. Laporte, "Arc Routing Problems. Part II: The Rural Postman Problem," *Operations Research*, 43 (1995), 399-414.

Fleischner, H., *Eulerian Graphs and Related Topics (Part 1, Volume 1), Annals of Discrete Mathematics*, 45, North-Holland, Amsterdam, 1991.

Ford, L. R. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, New Jersey, 1962.

Frederickson, G. N., "Approximation Algorithms for Some Postman Problems," *Journal of the Association for Computing Machinery*, 26 (1979), 538-554.

Frederickson, G. N., M. S. Hecht, and C. E. Kim, "Approximation Algorithms for Some Routing Problems," *SIAM Journal on computing*, 7 (1978), 178-193.

Friden, C., A. Hertz and D. de Werra, "STABULUS: A Technique for Finding Stable Sets in Large Graphs with Tabu Search," *Computing*, 42 (1989), 35-44.

Friden, C., A. Hertz and D. de Werra, "TABARIS: An Exact Algorithm Based on Tabu Search for Finding a Maximum Independent Set in a Graph," *Computers and Operations Research*, 17 (1990), 437-445.

Gendreau, M., A. Hertz, and G. Laporte, "New Insertion and Postoptimisation Procedures for the Traveling salesman Problem," *Operations Research*, 40 (1992), 1086-1094.

Gendreau, M., A. Hertz, and G. Laporte, "A Tabu Search Heuristic for the Vehicle Routing Problem," *Management Science*, 40 (1994), 1276-1290.

Gendreau, M., G. Laporte and R. Séguin, "Stochastic Vehicle Routing," *European Journal of Operational Research*, 88 (1996), 3-12.

Gendreau, M., P. Soriano, and L. Salvail, "Solving the Maximum Clique Problem Using a Tabu Search Approach," *Annals of Operations Research*, 41 (1993), 385-404.

Glover, F., "Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers and Operations Research*, 13 (1986), 533-549.

Glover, F., "Tabu Search - Part I," *ORSA Journal on Computing*, 1 (1989), 190-206.

Glover, F., "Tabu Search- Part II," *ORSA Journal on computing*, 2 (1990), 4-32.

Glover, F. and M. Laguna, *Tabu Search*, Kluwer, Massachusetts, USA, 1997.

Glover, F., E. Taillard, and D. de Werra, "A User's Guide to Tabu Search," *Annals of*

Operations Research, 41 (1993), 3-28.

Golden, B. L. and R. T. Wong, "Capacitated Arc Routing Problems," *Networks*, 11 (1981), 305-315.

Golden, B. L., J. S. DeArmon, and E. K. Baker, "Computational Experiments with Algorithms for a Class of Routing Problems," *Computers & Operations Research*, 10 (1983), 47-59.

Grötschel, M. and Z. Win, "A Cutting Plane Algorithm for the Windy Postman Problem," *Mathematical Programming*, 55 (1992), 339-358.

Guan, M., "Graphic Programming Using Odd and Even Points," *Chinese Mathematics*, 1 (1962), 273-277.

Guan, M., "On the Windy Postman Problem," *Discrete Applied Mathematics*, 9 (1984b), 41-46.

Haslam, E. and J. R. Wright, "Application of Routing Technologies to Rural Snow and Ice Control," *Transportation Research Record*, No. 1304 (1991), 202-211.

Held, M. and R. M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," *Mathematical Programming*, 1 (1971), 6-25.

Hertz, A. and D. de Werra, "Using Tabu Search Techniques for Graph Coloring," *Computing*, 29 (1987), 345-351.

Hertz, A., G. Laporte, and M. Mittaz, "A Tabu Search Heuristic for the Capacitated Arc Routing Problem," Working Paper, Département de Mathématiques, École Polytechnique Fédérale de Lausanne, December 1996.

Hertz, A., G. Laporte, and P. Nanchen, "Improvement Procedures for the Undirected Rural Postman Problem," Publication CRT-96-30, Centre de recherche sur les transports, Université de Montréal, August 1996.

Holland, J. H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, 1975.

Jaillet, P., Probabilistic Traveling Salesman Problem, Ph. D. Thesis, Report No. 185, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA, 1985.

Jaillet, P., "A Priori Solution of a Traveling Salesman Problem in Which a Random Subset of Customers are visited," *Operations Research*, 36 (1988), 929-936.

Jaillet, P. and A. R. Odoni, "The Probabilistic Vehicle Routing Problem," in *Vehicle*

Routing: Methods and Studies, B. L. Golden and A. A. Assad (eds.), North-Holland, Amsterdam, 1988.

Kappauf, C. H. and G. J. Koehler, "The Mixed Postman Problem," *Discrete Applied Mathematics*, 1 (1979), 89-103.

Laporte, G. and F. V. Louveaux, "The Integer L-Shaped Method for Stochastic Integer Programs with Recourse," *Operations Research Letters*, 13 (1993), 133-142.

Laporte, G. F. V. Louveaux, and H. Mercure, "A Priori Optimization of the Probabilistic Traveling Salesman Problem," *Operations Research*, 42 (1994), 543-549.

Larson, R. C. and A. R. Odoni, *Urban Operations Research*, Prentice-Hall, Engle-Wood Cliffs, New Jersey, 1981.

Lawler, E. L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart & Winston, New York, 1976.

Lemieux, P. F. and L. Campagna, "The Snow ploughing Problem Solved by a Graph Theory Algorithm," *Civil Engineering Systems*, 1 (1984), 337-341.

Lenstra, J. K. and A. H. G. Rinnooy Kan, "On General Routing Problems," *Networks*, 6 (1976), 273-280.

Levy, L. and L. D. Bodin, "Scheduling the Postal Carriers for the United States Postal Service: An Application of Arc Partitioning and Routing," in *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad (eds.), North-Holland, Amsterdam, 1988.

Levy, L. and L. D. Bodin, "The Arc Oriented Location Problem, *INFOR*, 27 (1989), 74-94.

Malek, M., A. Mourad, and M. Pandya, "Topological Testing," *Proceedings of the IEEE 1989 International Test Conference*, 1989, 103-110.

McBride, R., "Controlling Left and U-Turns in the Routing of Refuse Collection Vehicles," *Computers & Operations Research*, 9 (1982), 145-152.

Minieka, E., "The Chinese Postman Problem for Mixed Networks," *Management Science*, 25 (1979), 643-648.

Nobert, Y. and J.-C. Picard, "An Optimal Algorithm for the Mixed Chinese Postman Problem," *Networks*, 27 (1996), 95-108.

Orloff, C. S., "A Fundamental Problem in Vehicle Routing," *Networks*, 4 (1974),

35-64.

Osman, I. H., "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem," *Annals of Operations Research*, 41 (1993), 421-451.

Papadimitriou, C. H., "On the Complexity of Edge Traversing," *Journal of the Association for Computing Machinery*, 23 (19760, 544-554.

Picard, J. -C. and M. Queyranne, "On the structure of all Minimum Cuts in a Network and Applications," *Mathematical Programming Study*, 13 (1980), 8-16.

Picard, J. -C., and H. D. Ratliff, "Minimum Cuts and Related Problems," *Networks*, 5 (1975), 357-370.

Pearn, W. -L., "New Lower Bounds for the Capacitated Arc routing Problem," *Networks*, 18 (1988), 181-191.

Pearn, W.-L., "Approximate Solutions to the Capacitated Arc Routing Problem," *Computers & Operations Research*, 16 (1989), 589-600.

Pearn, W.-L., "Augment-Insert Algorithms for the Capacitated Arc Routing Problem," *Computers & Operations Research*, 18 (1991), 189-198.

Pearn, W. -L., A. A. Assad, and B. L. Golden, "Transforming Arc Routing into Node Routing Problems," *Computers & Operations Research*, 14 (1987), 285-288.

Potvin, J.-Y., T. Kervahut, B. L. Garcia, and J.-M. Rousseau, "The Vehicle Routing Problem with Time Windows – Part I: Tabu Search," *INFORMS Journal on Computing*, 8 (1996), 158-164.

Pureza, V. M., and P. M. França, "Vehicle Routing Problems via Tabu Search Metaheuristic," Publication CRT-747, Centre de recherche sur les transports, Université de Montréal, Canada, 1991.

Rego, C., and C. Roucairol, "A Parallel Tabu Search Using Ejection Chains for the Vehicle Routing Problem," in Meta-Heuristics: Theory and Applications, I. H. Osman and J. P. Kelly (eds.), Kluwer, Massachusetts, USA, 661-676, 1996.

Rochat, Y. and É. Taillard, "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing," *Journal of Heuristics*, 1 (1995), 147-167.

Rossi, F. A. and I. Gavioli, "Aspects of Heuristic Methods in the Probabilistic Traveling Salesman Problem," in Advanced School on Stochastics in Combinatorial Optimization, G. Andreatta, F. Mason and P.Serafini (eds.), World Scientific, Singapore, 214-227, 1988.

Roy, S. and J. -M. Rousseau, "The Capacitated Canadian Postman Problem," *INFOR*, 27 (1989), 58-73.

Sahni, S. K. and T. Gonzalez, "P-Complete Approximation Problems," *Journal of the Association for Computing Machinery*, 23 (1976), 555-565.

Semet, F. and É. Taillard, "Solving Real-Life Vehicle Routing Problems Efficiently Using Tabu Search," *Annals of Operations Research*, 41 (1993), *469*-488.

Stern, H. and M. Dror, "Routing Electric Meter Readers," *Computers & Operations Research*, 6 (1979), 209-223.

Swersey, A. J. and W. Ballard, "Scheduling School Buses," *Management Science*, 30 (1984), 844-853.

Taillard, É., "Some Efficient Heuristic Methods for the Flowshop Sequencing Problem," *European Journal of Operational Research*, 47 (1990), 65-74.

Taillard, É., "Parallel Iterative Search Methods for Vehicle Routing Problems," *Networks*, 23 (1993), 661-673.

Taillard, É., P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin, "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows," *Transportation Science*, 31 (1997), 170-186.

Turner, W. and E. Hougland, "The Optimal Routing of Solid Waste Collection," *AIIE Transactions*, 7 (1975), 427-431.

Ulusoy, G., "The Fleet Size and Mix Problem for Capacitated Arc Routing," *European Journal of Operational Research*, 22 (1985), 329-337.

van Aardenne-Ehrenfest, T. and N. G. de Bruijn, "Circuits and Trees in Oriented Linear Graphs," *Simon Stevin*, 28 (1951), 203-217.

Widmer, M. and A. Hertz, "A New Method for the Flow Sequencing Problem," *European Journal of Operational Research*, 41 (1989), 186-193.

Win, Z., *Contributions to Routing Problems*, Doctoral Dissertation, Universität Augsburg, 1987.

Win, Z., "On the Windy Postman Problem on Eulerian Graphs," *Mathematical Programming*, 44 (1989), 97-112.

Wunderlich, J., M. Collette, L. Levy, and L. D. Bodin, "Scheduling Meter Readers for Southern California Gas Company," *Interfaces*, 22 (1992), 22-30.