UNIVERSITY OF CALIFORNIA

Santa Barbara

On the Development of a New Class of Covering-Path Models

A Dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Geography

by

Timothy John Niblett

Committee in charge:

Professor Richard Church, Chair

Professor Konstadinos Goulias

Professor Erik Rolland, Cal Poly Pomona

Professor Stuart Sweeney

December 2016

The Dissertation of Timothy John Niblett is approved.

_____

Konstadinos Goulias

_____

Erik Rolland

_____

Stuart Sweeney

_____

Richard Church, Committee Chair

September, 2016

On the Development of a New Class of Covering Path Models

by

Timothy John Niblett

ACKNOWLEDGEMENTS

There are many people who have helped me toward completion of this dissertation. However, first and foremost I would like to thank my advisor, Dr. Rick Church, whose input, feedback, and mentorship are invaluable. Not only has Rick stimulated my mind and impressed upon me the value of looking at the world in an analytical context, Rick has been a mentor, friend, and trusted confidant. I have truly enjoyed working and learning with Rick and without Rick's support this dissertation would not be possible. I would also like to thank my committee members, Kostas Goulias, Erik Rolland, and Stuart Sweeney for their comments and support. I would also like to thank my brother, Matt Niblett. Matt introduced me to the world of Geography and has been extremely helpful, particularly with elements of coding. Without his help this dissertation would have been much more difficult and I am very grateful for all the support he has given me. Thirdly, I would like to thank my family and friends. Mom and Dad: thanks for taking the time to read through things and help me proof my dissertation and find errors and for always supporting me, even when times were rough. Lacey, thank you for putting up with me coming over to your place and hanging out with Matt; I'd also like to thank you for generously offering me many dinners and meals as we worked things out. To my wife, Yating, thank you for offering your input even when I didn't always appreciate it, thank you for always supporting me and for always encouraging me to do my best and keep raising expectations. To my friends and cohort, thank you for the support and stimulating conversation. Special thanks go to Wenwen Li and Ting Lei; without their help in coding and in ArcGIS things would have been so much more difficult. To my officemates Xin Feng and Carlos Baez, thanks for the conversations and good times.

VITA OF TIMOTHY JOHN NIBLETT
September 2016

EDUCATION

Bachelor of Arts in History, California Polytechnic State University, San Luis Obispo, June 2006 (cum laude)
Master of Arts in History, California Polytechnic State University, San Luis Obispo, June 2008
Master of Arts in Geography, University of California, Santa Barbara, June 2013
Doctor of Philosophy in Geography, University of California, Santa Barbara, September 2016 (expected)

PROFESSIONAL EMPLOYMENT

2008-16: Teaching Assistant, Department of Geography, University of California, Santa Barbara
2012-16: Instructor, Department of Geography, University of California, Santa Barbara
2010-14: Research Assistant, Department of Geography, University of California, Santa Barbara

PUBLICATIONS

Niblett, T. J. and Church, R. L. (2016) "The Shortest Covering Path Problem: A New Perspective and Model." *International Regional Science Review* Vol. 39, No. 1: 131 – 151.

Niblett, T. J. (2013) "The Maximal Covering/Shortest Path Problem Revisited: An Examination and Reformulation of the Problem to Allow the Elimination or Attachment of Sub-Tours." *Thesis: University of California, Santa Barbara*.

Church, R. L. and T. J. Niblett (2012) "Transit Route Design for Smaller Cities: Working Towards Sustainability." Final Report for University of California Transportation Center Grant awarded 2011-2012.

Niblett, T. J. (2008) "Chaos Among Order: the Impact of Internet Communities on Modern Society." *Thesis: California Polytechnic State University*.

Niblett, T. J. (2006) "The Stranglehold of Sprawl: A Case Study of Modesto and Its Environs." *Senior Thesis: California Polytechnic State University*.

In progress

Church, R. L. and T. J. Niblett. (2016) "TRANSMax II: an extended model for transit route optimization." To be submitted to *European Journal of Operational Research*.

Niblett, T. J. and R. L. Church. (2016) “Are we there yet? The Maximal Covering Shortest Path and Median Shortest Path Problems Revisited.” To be submitted to *Geographical Analysis*

AWARDS

Jack and Laura Dangermond Travel Grant, University of California, Santa Barbara, 2014-16
Dissertation Fellowship, University of California, Santa Barbara, 2013-15

FIELDS OF STUDY

Major Field: Geography

Studies in Operations Research and Location Analysis with Professor Richard Church

Studies in Transportation with Professor Konstadinos Goulias

Studies in Operations Research with Professor Erik Rolland

Studies in Spatial Statistics and Location Analysis with Professor Stuart Sweeney

ABSTRACT

On the Development of a New Class of Covering Path Models

by

Timothy John Niblett

The basis of this dissertation work stems from the fact that if one examines system route maps for many bus transit systems in U.S. cities, an interesting pattern emerges. Routes often utilize embedded loops to increase accessibility coverage of a system at the expense of adding a marginal amount of length to the overall path. Further, such routes frequently share a common corridor with respect to traveling in opposing directions, but they may depart spatially from each other in terms of direction. These departures in direction represent embedded loops that are traversed in only one direction. However, the literature has not explored this issue, and in fact, often discourages or outright prevents any loops from occurring whether they are loops that are traversed in both directions or traversed in only one direction. Furthermore, past research on covering path models has not accounted for travel in opposing directions, even when attempting to model transit lines. This is due in part to the roots of the covering path literature.

This dissertation presents an analysis of past work and from that defines several new problems that are 'loop agnostic' – that is, they neither prevent nor encourage the formation of loops in an optimal route, essentially a new class of covering path problems. Although several loop agnostic models are developed in this dissertation to better

represent the maximal covering shortest path problem, these models only capture one aspect of loop use. In the classic Maximal Covering Shortest Path problem, it is assumed that its use in transit will be traversed in both directions. Further, the classic formulation prevents most loops from occurring. A new form of this model is developed that allows loops to be part of a solution, whenever such loops provide an improvement in the objective function value. This model is called "loop agnostic" as the model neither prevents nor requires loops to be used in a solution. This means that a loop can be present as part of the path, as an out-and-back path or a more complex loop which visits several other nodes before returning to a previously visited node, or even as a 'lollipop' shaped route attached to the origin node or the destination node. If one assumes that the covering path can be traversed both in the outbound and inbound directions (which past work has done), any loops that are present will be traversed in both directions and is what we refer to as a bi-directional loop. When addressing the question of bi-directionality in real world systems it is possible that a loop is traversed in only one direction. Such "uni-directional" loops are formed whenever inbound/outbound paths diverge and can be observed in many transit system maps, like those of Bozeman, MT; Eau Claire, WI; and San Luis Obispo, CA. This dissertation also proposes a new problem, the Bi-Directional MCSP, and formulates two new models that account for travel based upon inbound and outbound path directions which allows for the use of shared arcs and uni-directional loops as well as bi-directional loops.

This dissertation also presents results from the application of these new models as well as a new heuristic to a hypothetical test network as well as a real world network from Richardson/Garland, Texas. Results demonstrate that loops are present in many

optimal solutions and that the route designs that utilize loop structures such as a ‘lollipop,’ ‘barbell,’ and ‘figure eight’ may well be superior to route designs that do not incorporate loops. This gives credence to the designs of virtually all transit systems in small and medium sized cities in the United States.

TABLE OF CONTENTS

Chapter 1

1.1 Introduction

In the late 1970s and the early 1980s two researchers – Slater (1980, 1981) and Current (1981) – defined the notion that some facilities should be represented as a path or a tree on a network. They reasoned that paths could form the basis for a structure or system that provides access or service to surrounding system elements. For example, Current suggested that a path might represent a transit line or a bus line that provides transit services to those living or working near the designed path. Whereas both Slater's and Current's research involved structures such as a path defined on a network or graph; Current modeled access or service based upon a maximal service/covering distance while Slater measured service based upon the sum of travel or access distances to the path or structure. This dissertation is concerned with the former construct where paths provide service coverage within a maximum range.

The initial groundbreaking model of Current (1981) involved finding the shortest path on a network connecting a prespecified origin node and a prespecified destination node, where the path travels sufficiently close to all other nodes of the network that all nodes receive access coverage. This service standard is formally defined as a maximal coverage distance or time, *S*. If a given node is within *S* distance or time of the path, then it is considered to be covered or served by the path. This original problem, in essence, involved finding the shortest path that provided coverage to all nodes and has been deemed the Shortest Covering Path problem (SCP). Current (1981) and Current et al. (1984) developed a model for this problem based upon an integer-linear programming

format. They proposed an iterative approach to solving this problem using linear programming with branch and bound at each iteration.

In subsequent work, Current et al. (1985) proposed lifting the condition that all nodes must be covered when proposing the Maximal Covering Shortest Path problem (MCSP). Their formulation for this problem was an extension of the original SCP model. The covering concept was then extended to Travelling Salesman Tour-based problems (Current and Schilling, 1989; 1994) and several other formulations and solution approaches have been proposed (Current, Pirkul, and Roland, 1994; Curtin and Biba, 2011; Wu and Murray, 2005; Matisziw et al., 2006; Boffey and Narula, 1998). With the exception of Curtin and Biba (2011), subsequent research involving the SCP and the MCSP problems and their extensions have been built on the basic model forms of the original papers of Current et. al. (1984 & 1985). Although Current and Biba formulated their model using a different format proposed in the Traveling Salesman Problem (TSP) literature, their model conforms to the basic premises underlying the original papers on SCP and MCSP.

There is an inherent assumption that has often not been stated explicitly when covering path model constructs are designed. This assumption deals with the direction of service. For example, if the path represents a transit route (Matisziw et. al., 2006; Wu and Murray, 2005; Current et. al., 1984, 1985; Murray and Wu, 2003), then it is assumed that the route is traveled in both directions. Thus, access is provided when traveling on the route whether heading towards one end of the path or towards the other end of the path. This seems to be a reasonable and consistent assumption as many transit routes typically involve bus or rail service along a path in both directions – e.g. a bus will travel in each

direction for a route located along a particular road. At first blush, it may be hard to think of any counter examples to this rule. The second inherent assumption in past work is that an optimal path will never loop back on itself. In fact, basic intuition seems to support the notion that if a path loops back and returns again to the same node, then that path is longer than needed. After all, why return to a node you've already been to? Clearly, such a circumstance seems to be inferior and therefore not optimal. For example, Curtin and Biba (2011) state that they formulated their TRANSMax transit model so that the path will not cross itself (and thereby produce a loop). In another example, Current et al. (1985) in formulating their maximum population covering path model include explicit constraints which stipulate that a node may be entered at most once along the path. This, too, is another form of preventing a path from looping back or crossing another part of the path.

Until the work of Niblett (2013) and Niblett and Church (2016), there has not been any question as to whether these two implicit, and sometime explicit, assumptions are true and hold when identifying optimal covering paths. Niblett (2013) and Niblett and Church (2016) demonstrated that for the original SCP, an optimal, complete covering path may involve the use of one or more loops/tours as an optimal solution. To do this they developed a new form of "sub-tour-breaking" constraints, a modified SCP model, and demonstrated how these new constraints could be used in an additive and iterative fashion similar to that of Dantzig et. al. (1954) and Current et al. (1984) to generate optimal solutions to an unrestricted SCP problem. This work clearly demonstrated that the implicit constraint imposed in the original SCP could not be supported when identifying and guaranteeing an optimal solution.

In this dissertation we extend the concept that optimal solutions may be comprised of loops/tours connected by a path or a path having attached loops/tours for the Maximal Covering Shortest Path problem. Three new formulations are presented for the following problems: the Maximal Covering Shortest Path problem, the Maximum Population Shortest Path problem, and the TRANSMax problem. In addition, a new heuristic algorithm is proposed for solving the unconstrained Maximal Covering Shortest Path problem and computational results are presented.

In a separate vein, this dissertation also explores the underlying assumption that a path represents a structure that is to be traveled in both directions. It can be observed that on public transit systems, a bus route or path may differ depending on which arcs are traversed which itself can be based upon the direction traveled. Sometimes this is done because the underlying network contains one-way roads and the bus route travels in parallel street segments depending on the direction of travel and street restrictions. Thus, the route may be separated by a city block but can otherwise be thought of as traveling in the opposite direction because it is not possible to traverse the same arc in the opposite direction. However, at other times it can be seen that travel along the two directions may be for the purposes of efficiently increasing access coverage. This can be observed in the map of routes used in the City of Bozeman, Montana given in Figure 1.1.

Each route contains loops and paths that may or may not be traversed in each direction. To model this type of design, this dissertation proposes a new path model called the Bi-Directional Maximal Covering Shortest Path problem (BD-MCSP). This problem involves determining a route composed of two path directions, one going from the origin to the destination (forward) which is the outbound path and the other going

from the destination to the origin (reverse) which is the inbound path. For both paths, a certain proportion of the route length involves arcs that are traversed in both directions (forward and reverse directions). This means that geographically, routes may be essentially the same in most respects for both directions, but deviations from the shared route structure are made in one direction or the other to efficiently increase coverage. Whereas the MCSP accounts for coverage and path distance in only one direction, this model accounts for coverage and path distances in both forward and reverse directions.

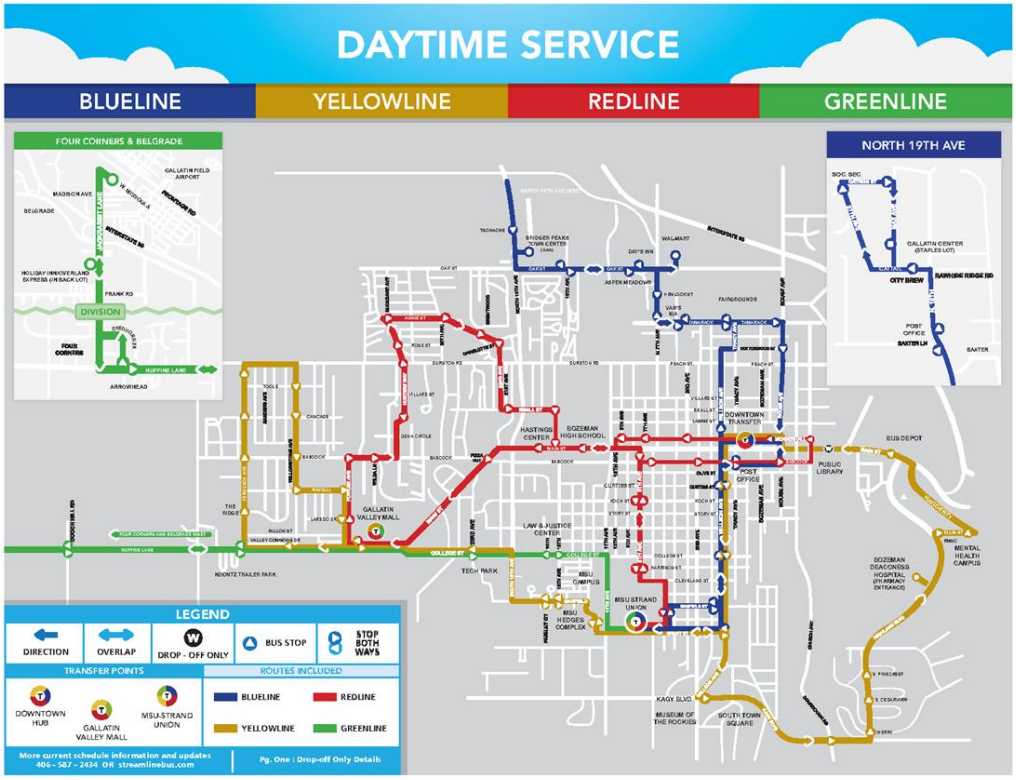


Figure 1.1 - Route Map for the City of Bozeman, MT. Permission to publish the system map has been generously given by the City of Bozeman.

The dissertation is organized as follows. Chapter 2 presents a review of the relevant literature. This includes a review of related models as well as several solution procedures and heuristics. Chapter 3 presents revised models for the MCSP and the

Maximum Population Shortest Path (MPSP) problems that overcome the restrictions inherent in the previous formulations of this model. This chapter also presents computational experience in solving this problem as well as several example solutions that demonstrate that this model can find better solutions than the original models of Current et al. (1985). Chapter 4 presents a new heuristic solution strategy for the revised MCSP. The focus of this heuristic is to generate 'good' solutions on large problems where the determination of optimal solutions takes a large amount of time. Chapter 5 addresses the Transit Arc Node Service Maximization Problem (TRANSMax) problem that was proposed by Curtin and Biba (2011). Their model restricts solutions from crossing or looping where the model form employed by Curtin and Biba is quite different from that of Current et. al. (1984, 1985). This chapter presents a new formulation for the TRANSMax problem – TRANSMax II – that can allow for crossing or looping while retaining the other features of their model. An application of this model is developed for the Garland/Richardson, Texas area. An example is shown where an embedded loop is a part of the optimal path/route. Chapter 6 presents details on a new problem called the Bi-directional Maximal Covering Shortest Path problem (BD-MCSP). This type of model accounts for travel in both forward and reverse directions – e.g. travel from a point A to a point B and the return (reverse) trip from point B back to point A. Several model formulations are developed and applied to example problems based upon the Swain dataset. We show that allowing flexibility in the design based upon route direction can improve service at little extra expense. Finally, Chapter 7 provides a summary of the contributions of this dissertation as well as the needs for future research.

Chapter 2

2.1 Introduction

This chapter will review the major research concerning covering paths up to the present. It is divided into the following sections: Key Problems, Extended Problems, Solution Approaches, and Concluding Remarks. Each section will examine key problems and topics with respect to the classic Maximal Covering Shortest Path problem and will discuss the related formulations as well as the broader impact of each topic. All model formulations will be given in a Mathematical Programming format. This means that each model will have an objective that is to be optimized as well as constraints that are necessary to meet pertinent conditions defining feasibility.

The Key Problems section presents several problems that are critical in the development of the Maximal Covering Shortest Path Problem. The Extended Problems section discusses related problems to the Maximal Covering Shortest Path problem and the broader impacts that they have. The Solution Approaches section outlines several methods that have been developed to solve covering path problems which will be useful for the development and implementation of the proposed heuristic in this dissertation. The final Concluding Remarks section is a brief overview of the papers and methods discussed as well as their relevance to the new formulations presented in this dissertation.

2.2 Key Problems

In order to trace the lineage of the Maximal Covering Shortest Path Problem one must look at several models that have been developed in the location science literature. These key models are as follows: the Traveling Salesman Problem, the Location Set Covering Problem, the Maximal Covering Location Problem, the Shortest Path Problem, the Shortest Covering Path Problem, and the Maximal Covering Shortest Path Problem. Each of these problems are presented in the context of covering models, path models, and their synthesis into covering path models. The last segment of the first section focuses on some nuances that need to be addressed in terms of *a priori* assumptions of past work.

The first key problem underpinning the research of this dissertation is the Traveling Salesman Problem (TSP). The goal of formulating and solving a Traveling Salesman Problem is to find the shortest route through a network such that each node (often representing a city) is visited at least once. This problem was defined nearly 200 years ago – the oldest known reference to a traveling salesman tour comes from a handbook published for traveling salesman routes in Germany and Switzerland in 1832.[1] However, the problem itself is rooted in the 7 Bridges of Königsberg Problem supposedly proposed to Leonhard Euler in the 1730's by the town fathers of Königsberg, Prussia. Euler was asked to find a parade route which would traverse the seven bridges of Königsberg once and only once. Euler proved that this was impossible to do in Königsberg's case, based upon a derived network representation and properties of this

[1] "Der Handlungsreisende – wie er sein soll und was er zu tun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein – von einem alten Commis-Voyageur" (The travelling salesman — how he must be and what he should do in order to get commissions and be sure of the happy success in his business — by an old commis-voyageur)

network. This work became the initial basis of the field of Graph Theory (Euler, 1741). The problem of determining a route which traverses every arc only once (as opposed to the TSP visiting every node) has become known as the Chinese Postman Problem due to the translation of the work of a Chinese mathematician, Kwan Mei-Ko, in the early 1960's.

The Traveling Salesman Problem, although having early practical interest, was found to be a very difficult problem to solve. Interestingly, the handbook from 1832 mentioned above presented a route that has more recently been found to be within 3% of an optimal solution – this can be seen in Figure 2.1 (Schrijver, 2003). However, it has been pointed out that if one takes into account the road conditions of the time, the solution offered in the handbook may even be the optimal solution! Nevertheless, although this solution comes very close to optimality, problems of this size remained computationally unsolvable to proven optimality until the advent of modern computers. The first modern mathematical work on routing involved school buses in

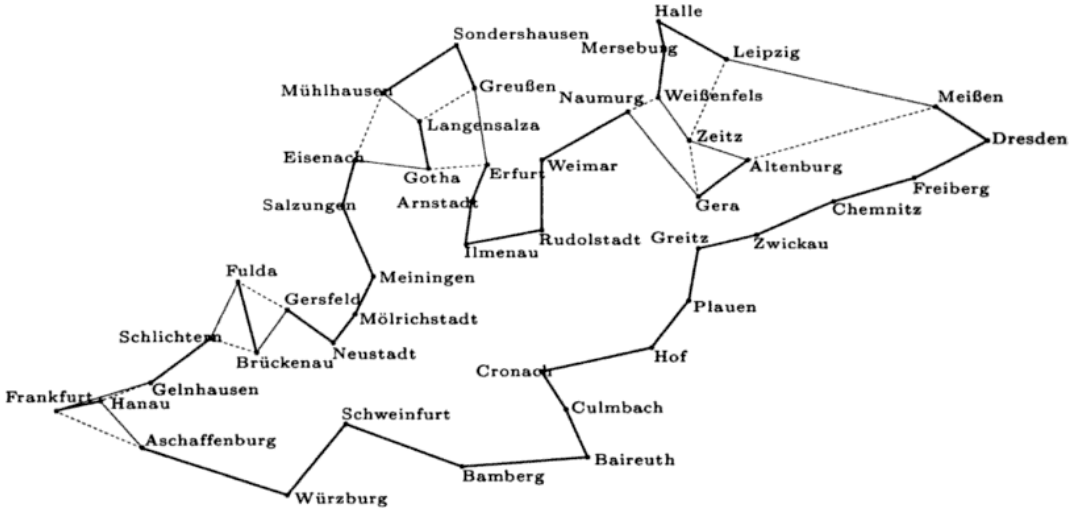


Figure 2.1 – Solution to a Traveling Salesman Route from 1832 with a Modern Comparison as found in Schrijver, 2003.

the 1930s (Dantzig, et. al., 1954). By the 1950's the advent of mathematical programming and the use of primitive digital computers finally allowed larger problems to be solved, and in 1954 Dantzig, Fulkerson, and Johnson formulated the first mathematical programming model. Since this is the first true mathematical formulation to be developed, and given that it is still used today (see Orman and Williams, 2006), it is given below as:

$i, j =$ indices used to reference the nodes of a given network

$c_{ij} =$ the cost of traversing the arc connecting node *i* to node *j*

$x_{ij} =$ decision variable for whether the arc connecting node *i* to node *j* is on the tour; its value is one if the arc is used and zero if not

$n =$ the total number of vertices (*v*) that comprise the network

$V =$ the set of all vertices/nodes

$S =$ represents a set of subsets of *V* in which it is possible to draw a sub tour

$$\text{Minimize } Z = \sum_{i \neq j} c_{ij} x_{ij} \quad \textbf{(2.1)}$$

Subject To

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 (i = 1,...,n) \quad \textbf{(2.2)}$$

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 (j = 1,...,n) \quad \textbf{(2.3)}$$

$$\sum_{v_i, v_j \in S}^{n} x_{ij} \leq |S| - 1 \ (S \subset V, |S| \geq 2) \quad \textbf{(2.4)}$$

$$x_{ij} \in \{0,1\} \quad (i, j = 1,...,n; i \neq j) \quad \textbf{(2.5)}$$

The formulation utilizes a cost array (c_{ij}) and a set of binary decision variables (x_{ij}) which are used to represent the arcs chosen for the tour and their associated costs; *S* represents the set of subsets of all vertices *V* in which it is possible to draw a sub tour.

The objective given in (2.1) minimizes the total cost for the tour. Constraint (2.2) ensures that, for each node *i*, one arc is used which 'leaves' that node. Constraint (2.3) ensures that at least one arc is used which 'arrives' at each node *j*. Constraint set (2.4) ensures that sub-tours cannot be a part of any solution. This type of constraint is referred to as a sub-tour breaking constraint. This set of sub-tour elimination constraints requires that for each subset *S* comprised of *V* vertices, the sum of the arcs used to connect the vertices *V* comprising that subset of *S* must be less than or equal to the total number of nodes in that set minus 1. Essentially, this constraint makes it impossible to have a complete loop between the nodes in the set *S*. Constraints of type (2.5) are simple binary constraints which restrict the value of each x_{ij} to either zero or one in value which indicate whether the arc from node *i* to *j* is not used or used in the solution.

This formulation is of critical significance for this dissertation for two primary reasons. The first reason is that, as noted above, the problem is one of the first formulations of a routing problem. However the second reason is quite significant with respect to the scope of the dissertation and that is due to the way this model has been solved. Although constraint (2.4) in the Dantzig, Fulkerson, and Johnson formulation specifies that all possible sub-tour sets should be prevented, the authors noted that this would be impractical and suggested a constraint adding strategy to solve the TSP. In essence this meant that one would solve a problem without constraint set (2.4), identify any sub-tours that appeared in the solution, define constraints of type (2.4) which would prevent these sub-tours from occurring, and then resolve the problem, and then repeat the process as necessary until no sub-tours are identified within the solution.

This process is significant because it has largely been carried over into the covering path literature, and is even suggested by Orman and Williams (2006) to be the most efficient method used to solve the TSP problem to optimality by mathematical programming. Although there is nothing inherently wrong with adding these sub-tour breaking constraints in an additive fashion in solving a TSP on a complete graph, there are nuances that should be considered when the method is applied in the solving the shortest covering path problem (Niblett and Church, 2016). The main contribution of this TSP formulation with respect to the covering path literature is that the sub-tour breaking constraints (2.4) are consistently used in the formulation of most models. A second element that is important to note is that this model was designed for a complete network. A complete network is one in which there is an arc between every pair of nodes (i.e. $n^2 - n$ arcs).

The second type of problem that is important to consider is the shortest path problem. The notion of traveling the shortest distance between one place and another has most likely occupied the mind of man from prehistoric times as they searched for shelter, water, or food. The first written reference of the shortest distances between places seems to be that of John Norden in 1607. The Norden map was the first gazetteer map which referenced not only places but also tried to convey topography and distinctive features along with a portion of local history.

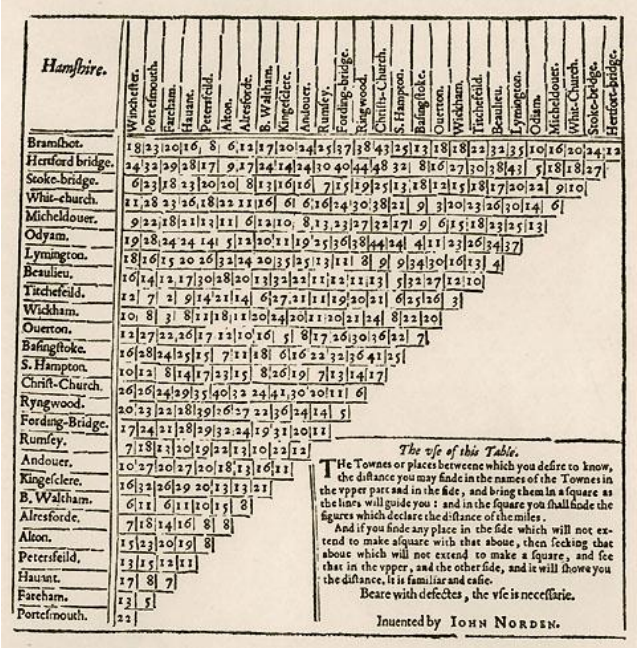


Figure 2.2 - Triangular Distance Table from John Norden's *England, an Intended Guyde for English Travailers* in 1625

By 1625, Norden updated his map to include a modern triangular distance matrix that one could use to determine the distance from one city to another within Hampshire, England which is shown in Figure 2.2. However, the mathematical inquiry into calculating the true shortest distance between points was not investigated until relatively recently; the first inquiries often dealt with routes through a maze and were done in the late 19th century (see Schrijver, 2012 for an excellent overview of these problems). In the late 1950's, as the advent of computer processing became more widespread, attempts were made to find the shortest route through algorithmic means. The Bellman-Ford algorithm was the first to appear (Bellman, 1956) although Dijkstra's algorithm is probably more well-known (Dijkstra, 1959).

In terms of a mathematical formulation, Orden (1956) and Dantzig (1957) were the first to describe the shortest path problem as an optimization model utilizing linear

programming.[2] Although Orden and Dantzig both formulated this problem as a linear programming model, the simplest form is that of Dantzig which will be presented here. The formulation is given as:

$A = \{i, j\}$ the set of arcs such that an arc connects node *i* and *j* in the network
$i, j =$ indices representing nodes on the network
d_{ij} = the shortest distance or time needed to travel from node *i* to node *j*
$x_{ij} =$ decision variable which is one if arc(*i*,*j*) is on the path and traversed from *i* to *j* and zero if not
$node\,1$ = the assumed starting node
$node\,n$ = the assumed terminus node
$N_j = \{i \mid arc(i, j) \text{ exists} \in A\}$

$$Minimize\, Z = \sum_{i \in N_j} \sum_{j} d_{ij} x_{ij} \qquad \textbf{(2.6)}$$

Subject To

$$\sum_{i \in N_1} x_{1i} = 1 \qquad \textbf{(2.7)}$$

$$\sum_{i \in N_n} x_{in} = 1 \qquad \textbf{(2.8)}$$

$$\sum_{i \in N_j} x_{ij} - \sum_{i \in N_j} x_{ji} = 0 \quad \text{for all } j;\ j \neq 1,n \qquad \textbf{(2.9)}$$

$$x_{ij} = \{0,1\} \qquad \textbf{(2.10)}$$

This formulation is based upon labeling each node with an index from 1, 2, 3, …, *n*. The indices *i* and *j* are used to refer to any specific node of the network. The above formulation is based upon the assumption that the arcs are undirected and that travel and distance in one direction on an arc need not be the same as the other direction. For each node *j* we can define a set of arcs that connect node *i* to its adjacent neighbors as N_i. The decision variables, x_{ij}, are used to specify whether a given arc is traversed in the direction of *i* to *j* in the shortest path. The objective (2.6) is to find the least cost path

[2] Dantzig mentions that his paper was based on a conference presentation made in 1955.

through the network. Constraint (2.7) ensures that one arc is chosen to leave the origin, that is, the path departs from the origin. In a similar manner, constraint (2.8) ensures that the path reaches the destination. Constraint (2.9) is a "balance" constraint that is included for each intermediate node (that is, all nodes except the origin and destination nodes). This constraint maintains that if an arc is used to enter a node, then an arc must be used to leave that node. Constraints of type (2.9) also maintain that if the path does not enter a node *j*, then it also must not leave that node. Constraint (2.10) is a binary variable constraint which insures that an arc is either used or not used in the shortest path. When this model is solved it will generate the optimal shortest path if one exists.[3] The concept for this formulation is relatively simple, but it is a critical component of the covering path literature in that the constraints given above (2.7-9) are used in virtually all covering path model formulations.

The other critical component one must have in order to create a covering path is a method to maintain coverage or define if it has been provided. In order to understand and follow the development of the covering path literature, we must then exam the models that have been formulated to cover demand nodes. Toregas et. al. (1971) are often attributed to be the first to propose a location problem to cover demands. Toregas et. al. (1971) proposed to find the smallest number of needed facilities and their locations such that all demands are covered. They defined that a demand point is covered if a facility is placed within a maximal service distance or time of that demand. They called this the Location Set Covering Problem (LSCP). The LSCP has been used as the basis for locating bus stops, emergency service facilities, and other public facilities such as

[3] A path will not exist when a network is disconnected with respect to its origin and destination.

libraries. The impact of this model in application has been quite extensive, particularly with respect to applications where complete coverage is of the utmost importance.

Toregas et. al. (1971) assumed that any point of demand is also a potential location point (although this assumption is not necessary) and that every point of demand must be covered (that is, it must be served within a maximal service distance or time denoted by *S*). The problem is formulated as follows:

x_i = 1 if a facility is allocated to site *j*, 0 if not

S = maximum distance/service time

d_{ij} = the shortest distance/time from node *i* to node *j*

N_i = {*j* in the set $J \mid d_{ij} \leq S$} the set of nodes within *s* distance/time of *i* that can provide acceptable service coverage

$$Minimize\, Z = \sum_{j=1}^{n} x_j \qquad \textbf{(2.11)}$$

Subject To

$$\sum_{j \in N_i} x_j \geq 1, \forall i \in I \qquad \textbf{(2.12)}$$

$$x_j = \{0,1\} \qquad \textbf{(2.13)}$$

The objective (2.11) in the LSCP is to locate the smallest number (i.e. the minimum number) of needed facilities x_j. Constraint (2.12) ensures that there is at least one facility that covers each demand point within the maximal service distance or time *S*. The members of each set N_i are calculated for every node and are dependent upon the maximal service distance, *S*. If *S* is set lower than the lowest d_{ij} (where $i \neq j$) then every site must be chosen for the solution, as no site will be able to cover any demand other than itself; whereas, if *S* is a very large number and the distances are significantly

smaller it may be possible that one site coves all demand. Constraints of type (2.13) are simple binary constraints which ensure that no fractional solutions are feasible.

In terms of coverage this is one of the first models proposed to optimally cover demand with public resources in mind. One can see how it would be of particular importance in that there are many services like fire protection, ambulance location, and of course rapid transit in which it would be ideal to cover an entire city with accessible services. In terms of applicability to a wide variety of public spatial problems, this formulation has been immensely useful. The drawback of this model is that when it is applied the cost of complete coverage may exceed the resources available. To address this limitation, Church and ReVelle (1974) proposed the Maximal Covering Location Problem. Instead of requiring complete coverage as in the LSCP, the MCLP involves maximizing coverage while limiting the number of facilities (i.e. fire stations, etc.) being used. Whereas the Location Set Covering Problem calls for coverage of all demand, the MCLP represents a relaxation of this requirement and can be used to generate a tradeoff between facilities used and coverage provided. The point on this tradeoff at which complete coverage is provided is a solution to the location set covering problem. Consequently, one can think of the MCLP subsuming the LSCP as a special case. Church and ReVelle (1974) formulated the MCLP as an integer-linear programming model using the following notation:

I = set of demand nodes
J = set of facility sites
S = distance beyond which there is no coverage
d_{ij} = the shortest distance from node i to node j
x_j = 1 if a facility is allocated to site j, 0 if not

N_i = { j in the set $J \mid d_{ij} \leq S$ }

a_i = population or other measure of demand at node i

p = total number of facilities to be located

$$Maximize\ Z = \sum_{i \in I} a_i y_i \quad \textbf{(2.14)}$$

Subject To

$$\sum_{j \in N_i} x_j \geq y_i \quad for\ all\ i \in I \quad \textbf{(2.15)}$$

$$\sum_{j \in J} x_j = p \quad \textbf{(2.16)}$$

$$x_j = (0,1) \quad for\ all\ j \in J \quad \textbf{(2.17)}$$

$$y_i = (0,1) \quad for\ all\ i \in I \quad \textbf{(2.18)}$$

In the MCLP formulation, we have an index of demand areas/points, *i*, and an index of facility sites, *j*. We also have a distance matrix d_{ij} which is the distance between a demand point *i* and a facility site *j*. There is a maximum service distance of *S* which is used to define the coverage sets N_i which list the set of sites which could provide cover to demand *i* should any site in the set be selected for facility placement. The objective function (2.14) of this model is to maximize the total demand covered. Site selection is represented by the decision variable x_j. The provision of coverage is tracked using the variable y_i which is 1 if demand *i* is covered and 0 if it is not. Constraint (2.15) allows node *i* to be counted as covered only when one or more facilities have been located which cover demand *i*. The second constraint restricts the total number of facilities used to be equal to *p*. Church and ReVelle noted that this problem can be set up as a minimization problem by minimizing what is not covered by

substituting $y_i = 1 - \bar{y}_i$ into the problem where $\bar{y}_i$ is 1 if *i* is not covered and 0 if it is.[4] If we then substitute in the variables, we would have the following objective function:

$$Maximize\ Z = (\sum_{i \in I} a_i + \sum_{i \in I} - a_i \bar{y}_i) \quad \textbf{(2.19)}$$

which then simplifies to:

$$Minimize\ Z = \sum_{i \in I} a_i \bar{y}_i \quad \textbf{(2.20)}$$

since the maximization of a negative number is the same as the minimization of the same positive number (Church and ReVelle 1974). Constraint (2.15) would also then be reformulated using the same variable substitution as (2.21):

$$\sum_{j \in N_i} x_j + \bar{y}_i \geq 1 \quad \textbf{(2.21)}$$

As mentioned above, one would be able to develop a cost benefit curve by solving for a range of values of *p* until a solution is obtained in which there is complete coverage. The flexibility of the MCLP in terms of being able to vary the number of facilities used for coverage has made this formulation quite useful and it has had many applications in a wide variety of spatial (and even some non-spatial) problems. In particular it has been used for fire station location, ambulance location (Schilling et. al., 1979; Daskin, 1983; Hogan and ReVelle, 1986, 1989; Marianov and ReVelle, 1996; Brotcome, et. al., 2003), mapping chaparral (Roberts, et. al., 1998), reserve site selection (Church et. al., 1996), humanitarian relief (Balcik and Beamon, 2008), selecting tooth color shades (Cocking, et. al., 2009), among many more spatial location applications.

[4] Setting the Maximal Covering Location Problem up as a minimization can often be desirable in order to suit the specific requirements of the problem being solved. One can computationally transform a problem as well by changing from a primal form into the corresponding dual. More on duality theory can be found in *Introduction to Operations Research* (Hillier and Lieberman, 1995).

Up to this point we have examined the Traveling Salesman Problem, the Shortest Path Problem, and the major location covering problems (LSCP and MCLP). We will now look at the fusion of covering and path problems in the form of the Shortest Covering Path problem (SCP) and the Maximal Covering Shortest Path problem (MCSP). These two problems each respectively represent a merging of the LSCP and the MCLP with the Shortest Path Problem. In terms of the covering path literature, Current et. al. (1984) were the first to propose the Shortest Covering Path Problem.

The Shortest Covering Path Problem involves finding the shortest path from an origin to a destination which passes within a maximal access distance/time of all nodes of the network. In the formulation proposed by Current et. al., the origin and destination nodes are also prespecified. Relevant applications of this problem include air delivery services, newspaper or other goods distribution, subway or other transit line creation, or even a way for a developing nation to determine where to upgrade infrastructure. The critical assumptions of the problem are as follows: 1) demand exists at every node, 2) all demands must be covered, 3) demands are covered if a node is directly on the path or is within the maximum service distance of a node on the path, 4) the system being modeled is uncapacitated, 5) all arc costs are non-negative, and 6) there are no budgetary constraints on the length/cost of the path. The ILP model is defined using the following additional or modified notation:

x_{ij} = 1 if the arc from *i* to *j* is on the shortest covering path and 0 otherwise

node 1 = the starting node for the shortest covering path

node n = the terminus node for the shortest covering path

S^* = the maximum allowable service distance/time

$S_k = \{ j | d_{jk} \leq S^* \}$ the set of nodes j which are within the maximum service distance to node k

$$Minimize\, Z = \sum_{i \in N_j} \sum_{j} d_{ij} x_{ij} \quad \textbf{(2.22)}$$

Subject To

$$\sum_{i \in N_i} x_{1i} = 1 \quad \textbf{(2.23)}$$

$$\sum_{i \in N_n} x_{in} = 1 \quad \textbf{(2.24)}$$

$$\sum_{i \in N_j} x_{ij} - \sum_{i \in N_j} x_{ji} = 0, \forall j \text{ where } j \neq 1 \text{ and } j \neq n \quad \textbf{(2.25)}$$

$$\sum_{i \in N_j} \sum_{j \in s_k} x_{ij} \geq 1, \forall k \text{ where } k \neq 1 \text{ and } k \neq n \quad \textbf{(2.26)}$$

$$x_{ij} = \{0,1\}, \forall (i,j) \quad \textbf{(2.27)}$$

In this formulation the objective (2.22) minimizes the total cost or length of the path, while ensuring all nodes are covered by the path. In terms of the SCP model constraints, Constraint (2.23) ensures that the path starts at a pre-specified starting node (given in the paper as node 1) and constraint (2.24) ensures that the path terminates at a pre-specified node (designated as node n). Each of these constraints specifies that only one arc may be used to leave the origin node and only one arc may be used to enter the destination node. These two constraints together then establish our origin and destination nodes respectively for the covering path. Constraint (2.25) is a flow balance constraint where if a path enters a node, it must exit that node. This constraint is written for all nodes except the origin and terminus nodes. This constraint ensures that the covering path is a sequence of connected arcs. Constraint (2.26) is the coverage constraint which requires that each demand node must be within S^* distance of a given arc comprising the covering path. The set S_k in constraint (2.26) represents the set of arcs that passes within a maximal distance S of a given demand node k, based upon the ending node of each arc

chosen for the path. The sum of the arcs that are used within this set has to equal or exceed 1 – in other words each demand must be covered at least once.

The idea behind the SCP formulation is that each demand node must be covered at least once and that the shortest possible path that does so must be found. However, when this model is solved on a network the solution that is obtained is likely to include what are called sub-tours. Sub-tours are a set of arcs that form a loop or an unattached tour which is not connected to the covering path (hence the sub-tour moniker). Technically speaking, these are the same elements that Dantzig et al. (1954) eliminated with their sub-tour breaking constraints in the Traveling Salesman Problem. The reason that these sub-tours form is that the model does not implicitly contain constraints to eliminate the so called sub-tours which may appear in a solution. Current et al. (1984) suggest that if sub-tours appear in a solution, the same sub-tour breaking constraints (2.4) as Dantzig et al. suggested should be employed and the problem re-solved. That is, the problem should be solved, the solution should be checked to see if any sub-tours exist, and if they do, add the associated constraints (2.4) which prevent these tours from forming and then the problem should be re-solved. This procedure is then repeated until no such tours appear in the solution. The reason that such sub-tours must be addressed is that they are anomalies that, unless prevented, will erroneously provide coverage away from the selected path due to the fact that they are 'unattached.' Niblett and Church (2016) noted, however, that the process for preventing these sub-tours must be properly applied as a Dantzig et. al. type constraint can impose conditions leading to a sub-optimal result.

In the covering path literature it has been assumed that, just as in the Traveling Salesman Problem literature, an attached loop or tour is undesirable. It should be noted that in the TSP literature it is assumed that all arcs form a complete graph, that is, a graph in which each node is adjacent (connected) to every other node. If one utilizes this kind of graph then it does indeed make sense to prevent any sub-tours. However, as one can easily observe on real world maps, it is impossible to go from one node directly to all other intersection nodes. Since we are searching for covering paths which, ostensibly, are going to travel on a real world network map, we are unlikely to utilize a complete graph and this key nuance will affect a covering path route if sub-tour adding constraints are not added properly (Niblett and Church, 2016). A detailed explanation of how this is addressed will be discussed later in this chapter. However, before we examine these nuances I want to first discuss the formulation of the Maximal Covering Shortest Path problem.

The Maximal Covering Shortest Path problem (MCSP) is a fusion of the Maximal Covering Location Problem (MCLP) and the Shortest Path Problem (SPP). In many ways it is a natural extension of the Shortest Covering Path problem (SCP) just as the MCLP is a natural extension of the Location Set Covering Problem (LSCP). This is due to the fact that, just as in the LSCP, the SCP problem requires that every demand node be covered. This means that there is less flexibility to determine tradeoffs with respect to coverage and overall path length/cost. Current et. al. (1985) formulated the MCSP such that these two competing objectives are addressed through the use of a multi-objective modeling framework. In this case, the MCSP involves two objectives; each weighted by an importance factor. The model formulation and notation is given below as:

a_k = population at k

α = weight associated with the coverage objective

$\beta = 1 - \alpha$ = weight associated with the distance objective

d_{ij} = the shortest distance/time from node i to node j

x_{ij} = one if the arc from i to j is on the shortest covering path

y_k = one if node k is covered and zero if it is not

node 1 = the starting node for the shortest covering path

node n = the terminus node for the shortest covering path

$N_j = \{ i | \text{arc}(i, j) \text{ exists} \}$ the set of nodes i which are connected to j

S^* = the maximum allowable service distance/time

$S_k = \{ j | d_{jk} \leq S^* \}$ the set of nodes j which are within the maximum service distance to node k

$$Z_C = \alpha \sum_{k=2}^{n-1} a_k y_k$$

$$Z_D = \beta \sum_i \sum_j d_{ij} x_{ij}$$

$$\text{Maximize } Z_C - Z_D \quad \textbf{(2.28)}$$

Subject To

$$\sum_{j \in N_j} x_{1j} = 1 \quad \textbf{(2.29)}$$

$$\sum_{i \in N_n} x_{in} = 1 \quad \textbf{(2.30)}$$

$$\sum_{i \in N_j} x_{ij} - \sum_{i \in N_j} x_{ji} = 0, \forall j \text{ where } j \neq 1 \text{ and } j \neq n \quad \textbf{(2.31)}$$

$$\sum_{i \in N_j} \sum_{j \in s_k} x_{ij} - y_k \geq 0, \forall\, k, k \neq 1, n \quad \textbf{(2.32)}$$

$$x_{ij} = (0,1), \forall (i,j) \quad \textbf{(2.33)}$$

$$y_k = (0,1), \forall\, k \quad \textbf{(2.34)}$$

In the model representation, the summation of $a_k y_k$ in the objective function (2.28) represents the total of what is covered with the exception of the starting and ending nodes – node 1 and node *n* respectively. If y_k is 1 then the associated population a_k is counted as being covered. In this objective, maximizing the first term represents maximizing the demand covered by the path. Similarly there is a competing objective in which we want to keep the path as short as reasonably possible. The overall length of the path is represented as the sum of the $d_{ij} x_{ij}$ terms. If arc x_{ij} is utilized in the solution $(x_{ij} = 1)$, then the associated distance, d_{ij}, is included. The sum of the associated distance/cost values, d_{ij}, corresponding to selected arcs, x_{ij}, yields the path length. When multiplying these values by a negative one (e.g. -1), the objective is reversed and is equivalent to minimizing path length. Altogether the objective is a composite of maximizing path coverage and minimizing path cost/length where each objective is weighted by importance factors α and β. Current et. al. (1985) formulated the coverage objective such that $k = 2$ to *n*-1 in order to account for the fact that the starting and ending nodes are explicitly specified in the problem and are automatically covered. One could, however, sum from *k* = 1 to *n* and include the starting and ending nodes and obtain the same covering path solution. Ignoring these two nodes in coverage does not impact the solution at optimality and requires two fewer variables – y_1, y_n – and two fewer constraints.

The model constraints for this formulation are similar to those we have seen earlier in this section due to the fact that this model formulation is a fusion of the MCLP and the SPP. In this model formulation, Constraint (2.29) ensures that only one arc on

the path leaves the origin node and constraint (2.30) ensures that only one arc on the path arrives at the destination node. Taken together these constraints ensure that a path has a specific origin and destination. Constraints of type (2.31) require that, for each intermediate node, if an arc on the covering path enters that node then there must also be an arc on the covering path which leaves that node. The reason for excluding the origin and destination nodes in constraint (2.31) is that Constraints (2.29) and (2.30) handle the specific case of the origin and destination. If the origin and destination nodes were not excluded we would have an ill-defined model. Constraints of type (2.32) are a modified form of the coverage constraint (2.15) in the Maximal Covering Location Problem. In this case, constraints of this type are defined such that variables which represent arcs instead of specific node locations are used. Specifically, if an arc is used which passes within S^* distance/time of node k then node k is covered and y_k can equal 1. Constraints of type (2.33-34) are simple binary constraints which are included in order to prevent fractional variable values that do not conform to the problem description.

Since this problem is multi-objective in nature, having an objective for coverage and an objective for path distance, this model may result in a number of different non-inferior/Pareto optimal solutions, each uniquely defined by how much emphasis is placed on one objective versus the other. We can identify those solutions which lie on the pareto-optimal trade-off curve by modifying the objective function. This is done through the use of the weighted terms α and β; a plot of these solutions obtained in this fashion can then be used to display the tradeoff curve. We can accomplish this by the use of the following modified composite objective:

$$\text{Maximize } Z = (\alpha)\sum_{k=2}^{n-1} a_k y_k - (\beta)\sum_i \sum_j d_{ij} x_{ij} \quad \textbf{(2.35)}$$

where α and β represents weight values that can vary from zero to one. Conversely, we can define $\alpha + \beta = 1$ without any loss of generality. This will allow us to make easily comprehensible weight values. For example, if we emphasize coverage (α) as 0.6 then the emphasis on overall path length (β) would be 0.4 which means that we would value coverage more than path length. Optimal solutions can be generated by solving the model for a range of values α and β. The results can be plotted to view the tradeoff between coverage and overall path length.

Current et al. (1985) also note that the MCSP can be modified via some structural adjustments into a problem defined as the Maximum Population Shortest Path, or MPSP, problem. This problem is defined for the case when the maximum coverage distance, *S**, is zero. This, in turn, implies that in order to be covered, a node must be a part of the path rather than within some distance of the path. This allows the objective function to be recast as:

$$\text{Maximize } Z = \sum_{i \in N_j} \sum_{j=2}^{n-1} a_j x_{ij} - \sum_{i \in N_j} \sum_j d_{ij} x_{ij} \quad \textbf{(2.36)}$$

Constraints (2.29-31) and (2.33) remain the same and constraint (2.32) is simplified to:

$$\sum_{j \in N_j} x_{ij} \leq 1, \forall j,\ j \neq 1 \quad \textbf{(2.37)}$$

which requires that no more than one arc may enter any given node (Current et al. 1985). This change reduces the number of variables in the problem which in turn reduces the

overall problem size. The change also has the effect of making the problem more akin to a TSP in that each node can only be visited once. As noted above, however, this may cause problems as a loop or return trip may be required to cover every node which would require a node appearing more than once in the route. This feature is expressively eliminated by constraints (2.37). Additionally, both the MCSP and its modified counterpart, the MPSP, suffer from the same issues as the SCP with respect to the way the problems are solved. Just as in the SCP, sub-tours are likely to occur in the solution if one solves the model as formulated. Thus, there may be one or more of these loops, or sub-tours, which exist independent of the covering path. In order to solve the problem to meet the stated criteria of having one contiguous covering path, that is one continuous path without any loops or sub-tours, a series of sub-tour breaking constraints (2.4) must be formed and added to the model and the model re-solved a number of times until no sub-tours exist in the resulting solution.

In solving the model, one needs to check the results and determine if any sub-tours are present. If it is determined that any sub-tours occur in the solution, the model is amended to prevent them by adding constraints for each sub-tour and then the problem is re-solved just as in the SCP.[5] The procedure for doing so is the same as that described for the SCP problem and is borrowed from the work of Dantzig, Fulkerson, and Johnson (1954). We must keep in mind that the DFJ sub-tour elimination process was developed within the context of a complete graph while the SCP and the MCSP/MPSP have been defined in the context of an incomplete graph. This nuance may actually result in an

[5] The authors note that since this model is set up as a bi-objective problem, there is a discrete set of non-inferior solutions which are called Pareto optimal. For more information please refer to *Multi-Objective Programming and Planning* by J. L. Cohon.

infeasible solution for a given problem even when a true feasible solution exists. The MPSP as formulated and solved by Current et. al. (1985) is particularly susceptible to this kind of issue although a similar result could occur on any of the covering-path type models given the right set of network conditions (See for example Figure 2.3). Niblett and Church (2013) were able to show that by utilizing the constraint structure developed by Dantzig, Fulkerson, and Johnson (1954) the optimal solution to a problem may, in fact, be overlooked. They also note that the work of Current et al. (1984 and 1985) expressively excludes the possibility of having any attached loops. The underlying assumption is that a loop/tour in the path is both undesirable as well as not optimal. Thus, it was thought that any loops, attached or otherwise, should be prevented. Although this approach seems reasonable and accurate, it is flawed and can be explained using an example of Niblett and Church (2016).

The sub-tour breaking constraint utilized in the process developed by Dantzig, Fulkerson, and Johnson in 1954 and used in the constraint additive process developed by Current et. al. in solving the covering path takes the form:

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij} \leq |V| - 1 \qquad \textbf{(2.38)}$$

where V is the set of vertices in the sub-tour and F_i represents the set of nodes that are reachable from node i – in other words it is the set of nodes that are reachable by an arc from node i. In general, there is one constraint for each vertex subset V. For a given set of V, the constraint represents the sum of the arcs connecting this subset of nodes, which must be strictly less than the number of members in V. Since the actual number of constraints of this form are too large to handle in a model and since many are not

necessary, we add such a constraint whenever we encounter the condition that violates the above condition for subset V and involves $|V|$ arcs in the solution. When such a case occurs, there will be a sub-tour or loop. However, as mentioned above, if one follows the DFJ constraint additive process starting with an incomplete graph it is possible that the true optimal solution may be prevented from being considered feasible. To

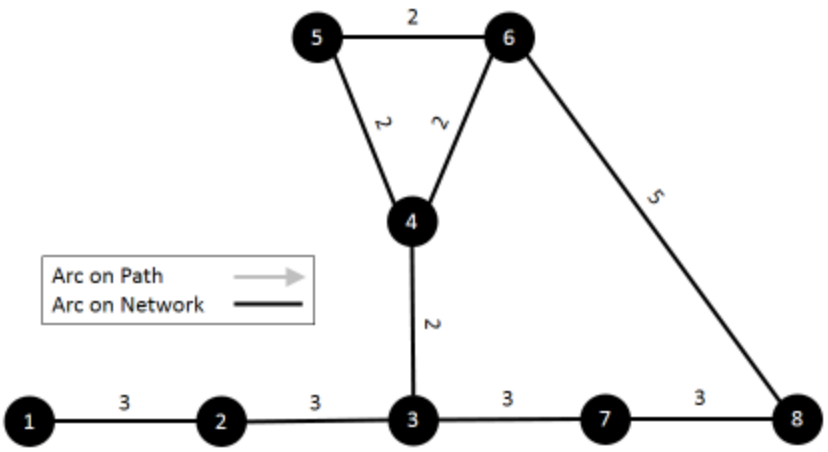


Figure 2.3A - An eight node, nine arc network. All arcs are undirected and can traverse in either direction. Costs are given as numbers above arcs.

illustrate this, consider the following example graph shown in Figure 2.3A. In this incomplete graph we have eight nodes and nine arcs which form an undirected graph. If we were to apply the SCP model[6] with a service distance of zero on this graph we would obtain the result seen in Figure 2.3B. Following the Dantzig, Fulkerson, and Johnson sub-tour breaking process we need to add a constraint to eliminate the cycle or sub-tour 4→5→6→4. After we add such a condition and re-solve, we generate the solution given in Figure 2.3C. The new solution determined has a path length of 23. However, is this really the optimal solution? In Figure 2.3D we can see that there is in fact a better

[6] The MCSP/MPSP could have been used as well though for illustrative purposes and the simplicity of the model formulation the SCPP is used here.

solution; involving an overall path length of 22. This clearly demonstrates that the sub-tour elimination constraint process developed by Dantzig, Fulkerson, and Johnson and applied by Current et. al. can prevent an optimal solution from being identified.

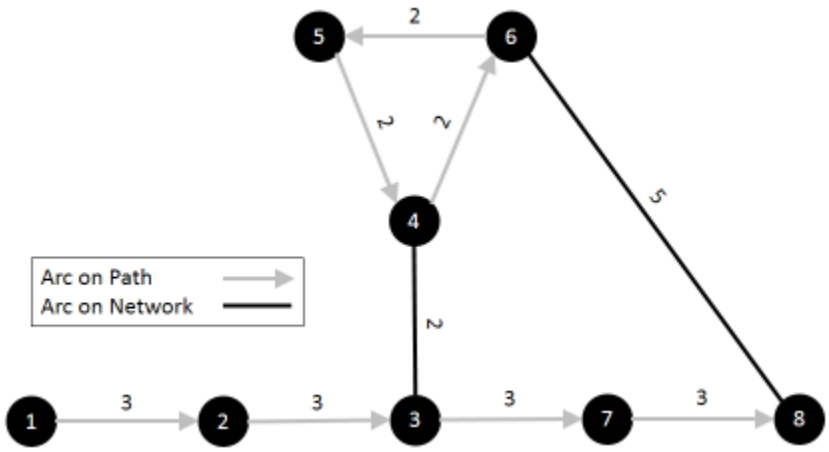


Figure 2.3B - An initial solution generated when solving the SCPP without constraints of type (7)

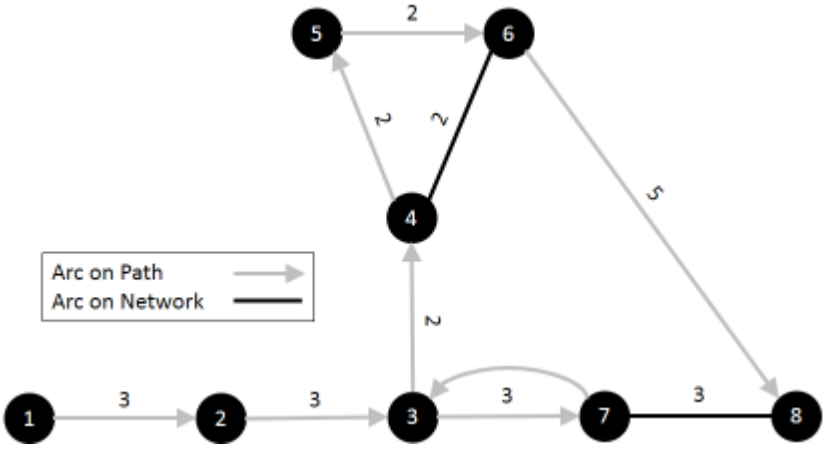


Figure 2.3C – The optimal solution to the SCPP after following the Dantzig, Fulkerson, and Johnson sub-tour elimination process with a path length of 23. Note that there is an attached tour. If this tour was forced not to exist there would be no feasible solution to the problem.

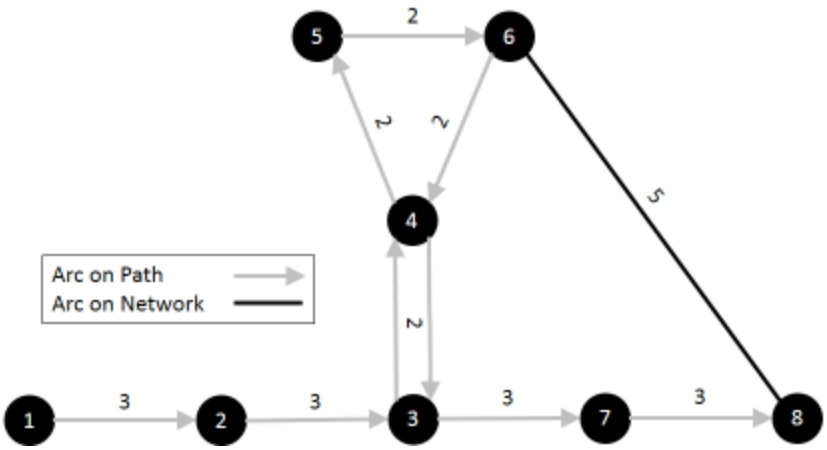


Figure 2.3D - The true optimal solution to a SCPP with a coverage distance of 0. The overall path length is 22.

To overcome this problem, Niblett and Church (2016) proposed a new type of constraint that forces a loop/sub-tour to be eliminated (as in Dantzig, Fulkerson, and Johnson) or attached to the path. This new condition is called an EAST constraint (Eliminate or Attach Sub-Tour). This new type of constraint developed by Niblett and Church (2016) takes the following form:

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij} - \sum_{i \in V} \sum_{\substack{k \in T_i \\ k \notin V}} x_{ki} \leq |V| - 1 \qquad \textbf{(2.39)}$$

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij} - \sum_{i \in V} \sum_{\substack{k \in F_i \\ k \notin V}} x_{ik} \leq |V| - 1 \qquad \textbf{(2.40)}$$

These two constraints taken together stipulate that in order for a tour/loop to exist in a solution it must have at least one arc enter and one arc leave the sub-tour to some other node which is not in the set *V*. The need to have an entering arc and an exiting arc for the sub-tour ensures that the cycle is attached elsewhere. Constraints of type (2.39) are used to ensure that if a sub-tour/loop is used in the solution it must have an external arc which is used to enter, or attach to, the sub-tour/loop. The first part of the mathematical

statement is a conventional DFJ based summation that sums the arcs used in the sub-tour denoted by the vertices in *V*. The new addition to this type of constraint is the term:

$$-\sum_{i \in V} \sum_{\substack{k \in T_i \\ k \notin V}} x_{ki} \qquad \textbf{(2.41)}$$

which represents arcs that can be used to enter the sub-tour from some node *k* such that $k \notin V$. Thus, a loop connecting nodes in *V* can occur when another arc is used in the solution to enter the loop/tour from some node which is not in *V*. If an arc is not used to enter the loop/tour from outside of *V* then the constraint prevents the loop/tour from being used.

Constraints of type (2.40) work in a similar fashion to constraints of type (2.39); the difference is that this type of constraint ensures that there must be at least one arc that leaves the sub-tour/loop. The second term of the constraint in (2.40):

$$\sum_{i \in V} \sum_{\substack{k \in F_i \\ k \notin V}} x_{ik} \qquad \textbf{(2.42)}$$

represents the sum of all arcs which are used to depart the sub-tour to some node *k* such that $k \notin V$ The summation given in (2.42) includes all decision variables x_{ik} that leave the sub-tour/loop within the set *V*. If the decision variables included in the summation given in (2.42) are all zero then the constraint given in (2.40) will revert into a Dantzig, Fulkerson, and Johnson style constraint that will prevent the sub-tour/loop from being formed. If one or more of the variables in (2.42) are equal to one, then the solution will have an arc which leaves the sub-tour in route to other nodes $k \notin V$ and allow the loop to

exist. Thus, taken together these constraints will either Eliminate or Attach a Sub-Tour/loop.

Once sub-tours within a solution have been identified and EAST constraints created, the problem must be re-solved. The process is then repeated; that is, we identify any sub-tours which have formed, create associated EAST constraints, and re-solve until no further sub-tours are identified. Therefore, if one utilizes this process, one can correctly identify the true optimal solution to any class of covering-path type of problem. The EAST constraint allows loops to occur within a solution, but does not require them. Whenever an optimal route is found by the use of this approach that involves the use of a loop, the original model would result in a solution that was not optimal.

We have now examined the key portions of the literature which have led to the development of covering-path type problems as well as looked at subtle but critical nuances to the way this class of problems is solved. We described portions of the Traveling Salesman Problem (TSP) literature as they have been applied in covering path models as well as examined the reasons for why loops/sub-tours were considered to be undesirable. We have also looked at the seminal problems in the covering science/location literature in the form of the Location Set Covering Problem (LSCP) and the Maximal Covering Location Problem (MCLP). The origins of the Shortest Path Problem (SPP) were then reviewed with particular attention paid to how the LSCP and MCLP formulations were merged into the seminal covering-path formulations of the Shortest Covering Path (SCP) and Maximal Covering Shortest Path problems (MCSP). Finally, we have reviewed how the traditional method of solving these problems needs to be amended in order to properly account for the use of loops/tours which can yield a true

optimal solution. In the next section, we will examine how the SCP and MCSP formulations have been utilized in applications.

2.3 Extended Problems

In this section several extended problems are discussed that are related to covering-path problems. These include the following problems: The Median and Maximal Covering Tour problems, the Covering Salesman problem, the Hierarchical Network Design problem, the Transit Arc-Node Service Maximization problem (TRANSMax), the Minimum Covering Shortest Path problem, and several multi-path covering problems. Just as in the previous section each problem will be reviewed and the modeling formulation will be defined and briefly discussed.

The Median and Maximal Covering Tour problems are a branch of the class of covering-path problems. The formulations for these models were developed by Current and Schilling (1994). The Median Tour Problem and Maximal Covering Tour problems differ from the SCP/MCSP in that, rather than find a covering path which travels through a series of intermediate nodes between a defined origin and destination node, we have a tour which can be thought of as beginning and ending at the same node. The goal of the Median Tour Problem is to visit p of n nodes such that the total tour length as well as the total travel distance necessary for all demand nodes to reach their nearest facility, or stop, on the tour is minimized. In effect it is a fusion of a Traveling Salesman Problem (TSP) together with a p-Median Problem. The goal of a p-Median Problem is to minimize the weighted distance from all demand nodes while locating p facilities (See Hakimi, 1964 and Maranzana, 1964). Hakimi defined the median location problem while Maranzana

was the first to propose a heuristic solution process. Although elements of the Traveling Salesman Problem are captured in the model formulation, a key distinction is that in both the Median and Maximal Covering Tour Problems the number of cities which are visited on the tour are decided *a priori*. That is, in a TSP one must visit all nodes in a given network while in the MTP/MCTP exactly p nodes must be visited on the tour. The formulation for the Median Tour Problem is given as:

x_{ij} = 1 if the arc from node i to node j is on the tour and 0 otherwise

y_{ij} = the fraction of demand at node i assigned to a facility/stop at node j on the tour

c_{ij} = the cost of including the path connecting nodes i and j on the tour

d_{ij} = the travel distance of the shortest path connecting node i to node j

a_i = the demand at node i

S = any subset of N

$|S|$ = the cardinality of set S

p = the number of stops/facilities on the tour

$$Minimize \quad Z = \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} + \sum_{i \in N} \sum_{j \in N} a_i d_{ij} y_{ij} \qquad \textbf{(2.43)}$$

Subject to

$$\sum_{j \in N} y_{ij} = 1 \quad \forall\, i \in N \qquad \textbf{(2.44)}$$

$$y_{ij} - \sum_{l \in N} x_{lj} = 0 \quad \forall\ j \in N \qquad \textbf{(2.45)}$$

$$y_{jj} - \sum_{l \in N} x_{lj} = 0 \quad \forall\ j \in N \qquad \textbf{(2.46)}$$

$$\sum_{i \in N} x_{ij} - \sum_{k \in N} x_{jk} = 0 \quad \forall\ j \in N \qquad \textbf{(2.47)}$$

$$\sum_{i \in N} \sum_{j \in N} x_{ij} = p \qquad \textbf{(2.48)}$$

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq |S| - 1 \quad \forall\, S \subset N \text{ such that } 2 \leq |S| < p \qquad \textbf{(2.49)}$$

$$x_{ij} \in \{0,1\} \ \forall\, (i, j) \qquad \textbf{(2.50)}$$

$$y_{ij} \geq 0 \ \forall\, (i, j) \qquad \textbf{(2.51)}$$

The MTP is a bi-objective problem. The first term in the objective is to minimize the overall tour length/cost. The second term of the objective involves the minimization of the total weighted distance for each demand accessing their nearest facility/stop on the tour. Since y_{ij} represents the portion of demand at node *i* assigned to node *j*, the 'cost' will be a function of the value of the demand at node *i*, a_i, as well as the distance from node *i* to node *j*, d_{ij}, times the value of y_{ij}. Constraints of type (2.44) ensure that all demands *i* must be assigned to a stop/facility *j* (e.g. $y_{ij} = 1$) that is a part of the tour. Constraints of type (2.45) require that all nodes to which demand is assigned must have an arc which is part of the tour enter the node to which demand is assigned in order for the demands to be served. Constraints of type (2.46) are self-assignment constraints which allow a demand to self-assign only if it is a node that is part of an arc on the covering tour. Constraints of type (2.45) and (2.46) taken together ensure service assignment to the tour is made for all demand. Constraints of type (2.47) are flow balance constraints which ensure that if a node is entered by an arc on the tour it must also have an arc on the tour which leaves that node. This constraint will then force a tour to form. Constraint (2.48) ensures that there will be *p* arcs used in the solution. Constraints of type (2.49) represent the Dantzig, Fulkerson, and Johnson sub-tour breaking style constraints. In this case, all sub-tours which have cardinality from 2 to *p*-1 are enumerated and prevented. These constraints are necessary as if they were not included, sub-tours often form in the solution which break the stated goal of finding one connected median tour. Constraints of type (2.50) are binary constraints which ensure that there are no fractional solutions. This will ensure that the entirety of an arc is either

chosen or not chosen. Constraints of type (2.51) are non-negativity constraints which prevent any negative demand assignment from occurring.

This formulation is quite useful in balancing tour length and total access distance. In fact, a tradeoff curve can be developed with respect to tour length and demand weighted distance costs by changing the values of *p*. Similarly, the use of objective weights could also be employed to change the emphasis placed on overall tour length/cost versus weighted demand coverage. However, a drawback of this formulation is that it requires a complete graph ($n^2 - n$ arcs) which in itself increases the size of the problem. An additional issue is the fact that a maximum distance/cost is not stated within the model. Another issue lies in the fact that this formulation can be thought of as using a coverage requirement akin to the Location Set Covering Problem where all demands must be covered. That is, all demands must assign to a facility which is part of an arc comprising the covering tour regardless of how far a given demand is from its closest facility on the tour. This may be acceptable in some cases, though in situations where a person or company is not likely to travel more than a certain distance the model would not adequately capture the problem. This is, in part, why the authors proposed a variant of the Median Tour Problem with the Maximal Covering Tour Problem formulation.

The Maximal Covering Tour problem objective is defined such that overall tour length is minimized as well as what is *not* covered by the tour. The key distinction between the Maximal Covering Tour Problem and the Median Tour Problem is that access coverage is maximized instead of minimizing the total distance of tour access. The additional set notation and formulation for the Maximal Covering Tour Problem is given as follows:

u_i = 1 if demand at node *i* is not covered by a stop/facility on the tour and 0 otherwise

$N_i = \{ j \mid d_{ij} \le s \}$ This is the set of all demands *j* which are covered by node *I* within the service distance *s*

s = the maximum service distance

$$Minimize \quad Z = \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} + \sum_{i \in N} a_i u_i \qquad \textbf{(2.52)}$$

Subject to

$$\sum_{i \in N} x_{ij} - \sum_{k \in N} x_{jk} = 0 \quad \forall \ j \in N \qquad \textbf{(2.53)}$$

$$\sum_{i \in N} \sum_{j \in N} x_{ij} = p \qquad \textbf{(2.54)}$$

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \le |S| - 1 \quad \forall \ S \subset N \text{ such that } 2 \le |S| < p \qquad \textbf{(2.55)}$$

$$\sum_{l \in N} \sum_{j \in N_i} x_{lj} + u_i \ge 1 \quad \forall \ i \in N \qquad \textbf{(2.56)}$$

$$x_{ij} \in \{0,1\} \ \forall \ (i, j) \qquad \textbf{(2.57)}$$

$$u_i \in \{0,1\} \ \forall \ i \qquad \textbf{(2.58)}$$

The first part of the objective (2.52) remains the same as in the Median Tour Problem; the cost of the tour path is minimized. However, the second term of the objective (2.52) is different. This second term of the objective minimizes what is not covered by the tour defined by the variables u_i. This variable, u_i, is one if demand *i* is not covered and zero if it is. The value of not covering demand *i* is given through the use of demand values a_i. Constraints (2.53) through (2.55) and constraint (2.57) are exactly the same as in the Median Tour Problem. Constraints of type (2.56) are used to tie the variable representing what is not covered, u_i, to the variables representing whether an arc is used as part of the covering-tour. Essentially, if no arcs on the tour pass sufficiently close to demand node *i*, then demand node *i* is not covered and u_i is forced to equal one in value. The last constraints (2.58) are the binary restrictions on u_i. Current and

Schilling (1994) note that the Maximal Covering Tour can be formulated as a Median Covering Tour problem through modification of the distance values d_{ij}. Transforming d_{ij} into d'_{ij} can be done as follows:

$$d'_{ij} = \begin{Bmatrix} 1, & \text{if } d'_{ij} > s \\ 0, & \text{if } d'_{ij} \leq s \end{Bmatrix} \qquad \textbf{(2.59)}$$

This means that the weighted distance in the objective function in (2.43) will represent the total demand not covered by the tour for a given service value *s*. However, utilizing the Median Tour Problem formulation to represent a Maximal Covering Tour will add more constraints to the problem than what is necessary in using the formulation of the Maximal Covering Tour problem given above.

The advantage of the Maximal Covering Tour model over that of the Median Tour problem is that it represents cases where one may wish to be within a certain standard such as overnight delivery, or in cases where a customer or a business will not travel farther than a certain distance in accessing service. The other advantage is that it has flexibility in not requiring coverage of all demand. This allows a business or service to design their service based upon a benefit/cost trade-off based upon the number of facilities/stops that are used and what the gain or loss of customers may be. This can be a powerful planning tool.

The next model of interest is the Covering Salesman Problem. This covering-path model is a fusion of the Traveling Salesman Problem and the Location Set Covering Problem. In a traditional Traveling Salesman Problem, all nodes must be visited using the shortest possible tour. In the Covering Salesman Problem, all nodes must be covered

within a maximum service distance utilizing the shortest possible tour. Although this is related to the Median Tour and Maximal Covering Tour problems it is different in that there is not a prespecified number of arcs that will be used to define which nodes are a part of the tour. Current and Schilling (1989) give the Covering Salesman formulation as follows:

x_{ij} = 1 if arc(i,j) is on the covering salesman tour and 0 otherwise

c_{ij} = the cost of using arc(i,j) in the covering salesman problem

Q = the set of solutions which exclude all solutions with sub-tours.

$P_l = \{ j \mid d_{ij} \le S_j \}$ = The set of nodes that are within the maximum coverage distance S_j

S_j = the maximum coverage distance for a stop at node j

d_{lj} = the shortest distance between node l and node j

$$Minimize \quad Z = \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \qquad \textbf{(2.60)}$$

Subject to

$$\sum_{i \in N} \sum_{j \in P_l} x_{ij} \ge 1 \quad \forall\, l \in N \qquad \textbf{(2.61)}$$

$$\sum_{i \in N} x_{ij} - \sum_{k \in N} x_{jk} = 0 \quad \forall\ j \in N \qquad \textbf{(2.62)}$$

$$\sum_{v_i, v_j \in S}^{n} x_{ij} \le |S| - 1 \ (S \subset \mathrm{V}, |\mathrm{S}| \ge 2) \qquad \textbf{(2.63)}$$

$$x_{ij} \in \{0,1\} \ \forall\ (i, j) \in N \qquad \textbf{(2.64)}$$

In this formulation, the objective (2.60) is to minimize the overall cost of the tour, just as it is in the Traveling Salesman Problem formulation. Constraints of type (2.61) ensure that each node is covered by at least one arc which is on the covering tour and which is within the maximal covering distance. This constraint is what allows the original TSP requirement that every node be visited once to be relaxed into the Covering Salesman requirement where every node must be either visited or be within a service range of a

node on the covering-tour. Constraints of type (2.62) are flow balance constraints which ensure that if an arc enters node *j* then a subsequent arc must also leave node *j*. Constraints of type (2.63) are given by the authors as constraints used to prevent sub-tours and take the form of a Dantzig, Fulkerson, and Johnson sub-tour breaking constraint. However, Current and Schilling (1989) note in their paper that since the set *Q* is impossible to know *a priori*, either a solution process similar to Dantzig, et. al. must be employed or an alternative sub-tour breaking constraint must be used. The final constraints (2.64) are the binary constraints which ensure that an arc is either used or not used and that there are no fractional solutions.

As was mentioned above, Current and Schilling use a sub-tour breaking constraint type which was developed by Gavish (1983) for use on the Capacitated Minimal Directed Tree Problem. In order to apply the method developed by Gavish, Current and Schilling noted that they needed to modify the original graph through the creation of a dummy node, defined as Node 0, and specifying that the cost of all arcs that go to and from Node 0 are equal to zero. Based on this form, constraints of type (2.63) can then be enumerated through the following:

$$\sum_{j \in N,\, j \neq 0}^{n} x_{0j} = 1 \qquad \textbf{(2.65)}$$

$$\sum_{k=0}^{n} x_{jk} = \sum_{i=0}^{n} x_{ij} + \sum_{i=0}^{n} z_{ij} \;\forall\; j = 1 \ldots n \qquad \textbf{(2.66)}$$

$$z_{j0} \leq (n+1) z_{0j} \;\forall\; j = 1 \ldots n \qquad \textbf{(2.67)}$$

$$z_{ij} \leq n x_{ij} \;\forall\; i, j = 1 \ldots n, \quad i \neq j \qquad \textbf{(2.68)}$$

$$z_{ij} \geq 0 \;\forall\; i = 1 \ldots n,\; j = 0 \ldots n, i \neq j \qquad \textbf{(2.69)}$$

$$z_{0j} \in \{0,1\} \; \forall \; j = 1 \ldots n \qquad \textbf{(2.70)}$$

The advantage of using the sub-tour elimination constraints (2.65-70) is that a model only needs to be solved once instead of using an iterative approach that repeatedly solves and adds constraints to a model until no sub-tours are part of the final solution.[7]

The way that the constraints above exclude a sub-tour from being part of the solution will now be described. Constraint (2.65) will ensure that one unit of flow leaves the dummy node represented as node 0. Constraints of type (2.66) are flow augmentation constraints. In this case, the flow out of *j* must be equal to the flow into *j* plus the number of arcs on the Covering Salesman tour which enter node *j*. Constraints of type (2.67) ensure that flow into the dummy node must originate from that node as the right hand side will be zero for all but the origin node. Constraints of type (2.68) ensure that flow will only occur on arcs that are used in the covering tour. Constraints of type (2.69) ensure that flow along all arcs is non-negative and constraints of type (2.70) ensure that the flow on all arcs from the dummy node must be either 0 or 1. Taken together, these constraint types and the complete graph in conjunction with a dummy node will ensure that the tour/circuit will not include a sub-tour or disconnected loop.

However, because these problems are a hybrid of location and Traveling Salesman Problems, they must use a complete graph. This means that as a problem increases in size, the problem begins to exponentially grow in terms of variables and constraints. This poses considerable challenges and limits the scope of the network size on which the problem can be optimally solved in a reasonable amount of time. Another

[7] Although there are many types of sub-tour elimination constraints such as Gavish and Graves (1978), Miller, Tucker, and Zemlin (1960), Vajda (1961), Lawler et. al. (1985), Orman and Williams (2006) suggest that an iterative approach may actually be a more efficient way of solving the problem.

drawback is that in effect you must use a complete network when in reality the network may be quite sparse. Interestingly, Current and Schilling (1989) note this in their paper where they show a solution to the CSP on an incomplete graph. This is readily seen in Figure 2.4.

This shows that the use of attached loops/tours as a covering strategy may in reality be an optimal solution. In terms of the modeling framework, one must use a complete graph with their model of sub-tour breaking constraints in order to find such embedded loops if they are optimal, whereas it is possible to use the EAST constraints of Niblett and Church (2016) on a sparse graph and find the same result more efficiently than the original approach. The next related problem that will be examined is the Hierarchical Network Design Problem. This problem was first proposed by Current, ReVelle, and Cohon (1986) and is formulated such that the least cost, two-level hierarchical network is to be found. That is, we wish to find a path between an origin

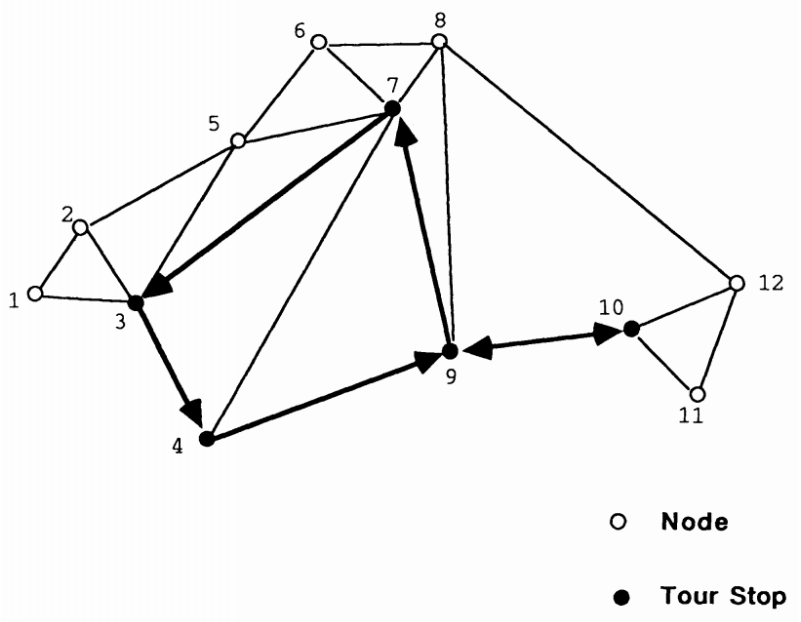


Figure 2.4 - A solution to the Covering Salesman Problem as shown on an incomplete graph (Current and Schilling, 1989).

and a destination as the primary path where all other nodes that are not part of the primary path are connected to the primary path via secondary paths. The objective is to minimize the length of the primary path as well as the lengths of all secondary paths. The costs of developing the primary path (dollar per distance) are assumed to be greater than that of secondary paths (dollar per distance). One can think of this type of hierarchical structure as a primary transit line with secondary feeder lines that originate in the hinterlands. Another way of thinking of this type of network would be for the allocation of resources. For example in developing countries, the primary path may represent a paved road between major populations while the secondary path would represent the location of dirt or gravel roads connecting smaller or isolated communities. Perhaps the most direct use would be in the application of power transmission lines where a primary route might be the location of a high voltage line and the secondary paths could be lower voltage distribution lines. An example of this type of solution is given in Figure 2.5.

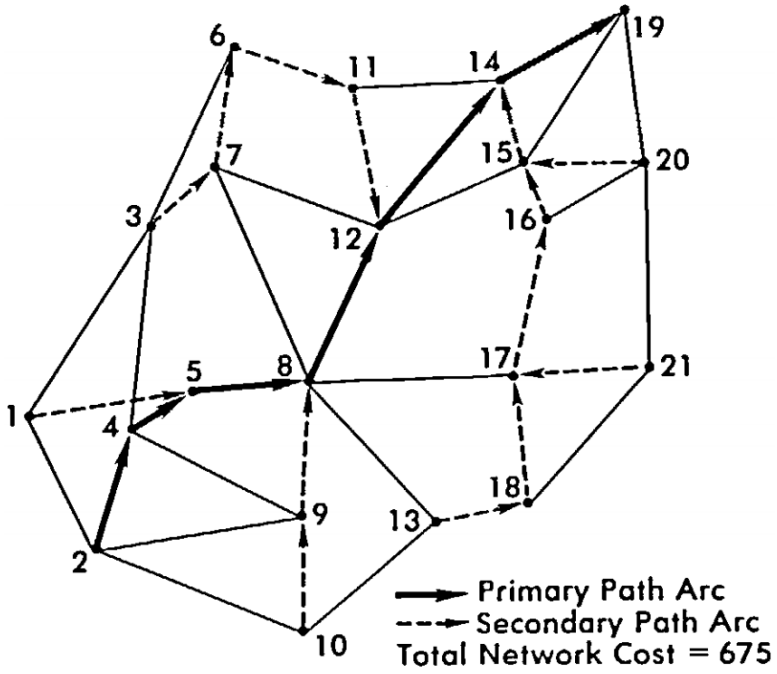


Figure 2.5 - An example solution to the Hierarchical Network Design Problem (Current, et. al., 1986).

When formulating this model the authors make the following assumptions: 1) demand exists at every node; 2) demand at every node must be satisfied; 3) demand at a node is satisfied if either the node is on the primary path or is connected to the primary path via a secondary path; 4) flow along all arcs is uncapacitated; 5) costs of trans-shipment facilities at the intersections of the two path types are negligible; 6) all are arc costs are non-negative; 7) there are no budgetary constraints. The formulation of the Hierarchical Network Design Problem is given below:

C_{ij} = the cost of a primary arc connecting node *i* to node *j*

$C_{ij}^{'}$ = the cost of a secondary arc connecting node *i* to node *j*

x_{ij} = 1 if a primary arc connects node *i* to node *j* and 0 otherwise

y_{ij} = 1 if a secondary arc connects node *i* to node *j* and 0 otherwise

$N_i = \{ j \mid \text{arc}(i, j) \text{ exists} \}$

$M_j = \{ i \mid \text{arc}(i, j) \text{ exists} \}$

Node *O* = the starting node

Node *D* = the terminus node

V = a subset of nodes

Q = the set of all nodes

|*Q*| = the cardinality of subset *Q*

$$Minimize \quad Z = \sum_i \sum_j c_{ij} x_{ij} + \sum_i \sum_j c_{ij}^{'} y_{ij} \qquad \textbf{(2.71)}$$

Subject to

$$\sum_{j \in N_O} x_{Oj} = 1 \qquad \textbf{(2.72)}$$

$$\sum_{i \in M_D} x_{iD} = 1 \qquad \textbf{(2.73)}$$

$$\sum_{i \in M_j} x_{ij} - \sum_{k \in N_j} x_{jk} = 0 \quad \forall \ j, \ j \neq O, D \qquad \textbf{(2.74)}$$

$$\sum_{j \in N_i} x_{ij} - \sum_{j \in N_i} y_{ij} = 1 \quad \forall\, i, i \neq O, D \tag{2.75}$$

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} - \sum_{i \in Q} \sum_{j \in Q} y_{ij} \leq |V| - 1 \quad \forall\, V \subseteq Q, \text{such that } |V| \geq 2 \tag{2.76}$$

$$x_{ij} \in \{0,1\}\ \forall\ (i, j) \tag{2.77}$$

$$y_{ij} \in \{0,1\}\ \forall\ (i, j) \tag{2.78}$$

The objective (2.71) minimizes the cost of the primary and secondary paths represented by x_{ij} and y_{ij} respectively. Constraint (2.72) ensures that one arc will leave the prespecified origin, *O*, on the primary path. Constraint (2.73) requires that one arc will enter the prespecified terminus node, *D*, on the primary path. Constraints of type (2.74) require that if a primary arc enters an intermediate node on the primary path, then an arc must also leave that intermediate node on the primary path. Constraints of type (2.75) require that a node must either have a primary arc or a secondary arc leave it with the exception of the origin and destination nodes. This type of constraint ensures that all nodes on the graph are connected by either a primary or a secondary path as it requires an arc to be used which is either part of the primary or secondary path. Constraints of type (2.76) are sub-tour elimination constraints derived from the Dantzig, Fulkerson, and Johnson constraint type given in (2.4). Constraint types (2.77) and (2.78) are binary constraints which force each decision variable to be either zero or one. As the complete set of sub-tours is not known *a priori*, the authors suggest a Dantzig, Fulkerson, and Johnson style iterative sub-tour elimination approach wherein one solves the model without any constraints (2.76), determines the existence of sub-tours, adds constraints of type (2.76) for each of the sub-tours involved in the solution, and then re-solves the problem repeating the process until no further sub-tours are involved in the solution. However, because this problem is solved on an incomplete graph, it is possible that the

sub-tour breaking constraints used in this type of problem would also need to be modified through the use of EAST constraints as it is possible that the optimal solution could be inadvertently excluded, just as in the SCP and MCSP problems.

The next problem formulation to be discussed is the Minimum Covering Shortest Path Problem developed by Current, et. al. (1988). This problem is a natural extension of the work done on the MCSP and SCP formulations. The problem itself is not complicated with the title itself descriptive of the problem; the goal is to find the shortest possible path which also covers as little as possible. A minimum covering shortest path modeling framework is particularly useful for applications such as corridor location, hazardous waste transportation, military frameworks wherein one wishes to avoid contact with the enemy, et cetera. In the case of a corridor location application, one may wish to avoid populations of a certain species or land areas which have a high ecological value. The movement of toxic materials and hazardous wastes has also become contentious in the public realm, particularly in light of the potential for attack after the events of September 11, 2001. Thus, finding the path of minimum impact/disruption is highly desirable from both a strategic, environmental, and public safety point of view.

With respect to the Minimum Covering Shortest Path Problem, the following assumptions are held: 1) flow along arcs is not capacitated; 2) Arc costs are non-negative; 3) There are no budgetary constraints; and 4) Population at a node is negatively impacted if the path comes within some predetermined impact/covering distance of that node. The formulation for the Minimum Covering Shortest Path problem is given as:

c_{ij} = the population covered by an arc from node *i* to node *j*

d_{ij} = the distance/time to travel on an arc from node *i* to node *j*

x_{ij} = 1 if an arc connects node *i* to node *j* and 0 otherwise

$N_i = \{ j \mid \text{arc}(i, j) \text{ exists} \}$

$M_j = \{ i \mid \text{arc}(i, j) \text{ exists} \}$

$$Minimize \quad Z = \sum_i \sum_j c_{ij} x_{ij} + \sum_i \sum_j d_{ij} x_{ij} \tag{2.79}$$

Subject to

$$\sum_{j \in N_O} x_{Oj} = 1 \tag{2.80}$$

$$\sum_{i \in M_D} x_{iD} = 1 \tag{2.81}$$

$$\sum_{i \in M_j} x_{ij} - \sum_{k \in N_j} x_{jk} = 0 \quad \forall \ j, \ j \neq O, D \tag{2.82}$$

$$x_{ij} \in \{0,1\} \ \forall \ (i, j) \tag{2.83}$$

where the objective function (2.79) minimizes the sum of what is covered by an arc as well as the total distance of the covering path. Constraint (2.80) ensures that the path will have one arc which leaves the origin node and constraint (2.81) ensures that the path will have one arc that arrives at the destination node. Constraints of type (2.82) are balance constraints which ensure that if an intermediate node, *j*, is entered by an arc from node *i*, then there must also be a corresponding arc that leaves node *j* and travels to another node, *k*. Constraints of type (2.83) ensure that the decision variables for arc use, x_{ij}, must be either 1 or 0 to represent whether an arc is used or not. Although Current et. al. do not formulate the problem as such, it is possible to include a neighborhood set in which one can define all nodes that are within a specified distance of the visited node. Such an addition would be useful if one wished to include geographic proximity of outlying areas into the model. Current et. al. state that in the formulation, population values at these nodes have been calculated a priori and incorporate surrounding areas into the population

set c_{ij}. The Minimum Covering Shortest Path problem could be solved using importance weights applied for each objective term, just as in the MCSP problem. This would allow one to determine a tradeoff curve based upon emphasis on minimizing coverage and minimizing path length.

Up to this point we have examined several related problems in the covering-path literature. These path problems have primarily utilized a tour or path and require the use of sub-tour elimination constraints to obtain solutions feasible to the stated problem. The next set of problems that we will examine are multi-path covering problems as well as a new transit route design model. The multi-path covering problems are extensions of the SCP/MCSP formulations. These include models such as the multi-path covering (Boffey and Narula, 1998), the multiple route transit network design (Wan and Lo, 2003; Wu and Murray, 2005), and the transit route extension (Matisziw, et. al., 2006) problems. Because of the unique use of Vajda constraints and the fact that the formulation involves a different modeling framework than the SCP/MCSP we will first examine the Transit Arc Node Service Maximization Problem (TRANSMax).

TRANSMax was recently developed by Curtin and Biba (2011); it was specifically designed for the development of transit lines and overcoming unique issues presented through the use of a GIS. It should be noted that there has been much work surrounding transit planning and routing. In transit planning there is a broad range of issues that must be considered. For example, one issue may be customer satisfaction as how a user perceives the safety, cleanliness, and cost (Levinson and Brown-West, 1984; Levinson, 1992; Weinstein, 2000; Figler, et. al, 2011; Tyrinopoulos and Antoniou, 2008). One of the first papers concerned specifically with transit was an adaptation of the LSCP

to locate bus stops (Gleason, 1975). However, much of the early literature from the mid 1970's to the late 1990s focused primarily on efficient allocation of resources (i.e. run cutting, headways, and scheduling) rather than optimally locating route alignments (Rousseau, 1985; Furth and Day, 1985; Ceder and Wilson 1986; Daduna and Wren, 1988; Wren and Wren 1995). In the mid-1990s and early 2000's an emphasis was placed on the issues of access and accessibility as well as how these goals could be implemented (O'Neill et. al., 1992; Hsiao, et. al., 1997; Murray, et. al., 1998; Murray, 2001, 2003). As computer processing power has improved over time, models were developed which could be optimally solved. The vast majority of routing models use the framework derived from Current et. al. (1984, 1985). Later in this section we will discuss several of these routing models, particularly Boffey and Narula (1998), Wan and Lo (2003), Wu and Murray (2005), and Matisziw, et. al. (2006). The key point of noting this is that only recently has attention been paid to optimal transit network design, and when this has been an issue the framework that is most often used is that of Current, et. al. (1984, 1985). This is what sets the TRANSMax model apart from others in the covering-path literature.

The TRANSMax model departs from the Current et. al. (1984, 1985) framework by implementing a model based upon Vajda's (1961) formulation of a Traveling Salesman Problem. Vajda's implementation utilizes a third index in addition to the traditional origin, destination indices for the routing decision variables. The additional index Vajda employs is used to denote the order an arc is used on the path. This ordering or sequencing is principally used in order to prevent the formation of sub-tours. The sub-tour preventing constraints developed by Vajda for the TSP take the form of (2.84):

$$\sum_{i=1}^{m} x_{ijt} - \sum_{i=1}^{m} x_{ji(t+1)} = 0 \quad \forall\ j, t = 1,2,\ldots,m \tag{2.84}$$

where m is the number of cities to be visited, i and j are the indices of cities/nodes, and t is the index representing the sequence of arcs along the route. Thus, in the constraint formulated by (2.84), if an arc in sequence t is used to travel from i to j then the t-th plus one arc must be used to travel from j to a subsequent node. To prevent sub-tours from forming the additional constraints given in (2.85), (2.86), and (2.87) are required.

$$\sum_{j=1,\, j\neq i}^{m} \sum_{t} x_{ijt} = 1 \quad \forall\ i \tag{2.85}$$

$$\sum_{i=1,\, i\neq j}^{m} \sum_{t} x_{ijt} = 1 \quad \forall\ j \tag{2.86}$$

$$\sum_{i} \sum_{j} x_{ijt} = 1 \quad \forall\ t \tag{2.87}$$

Constraints of type (2.85) are used to ensure that only one arc would be permitted to leave node i across all sequences t. Constraints of type (2.86) are used to ensure that only one arc is allowed to enter node j across all sequences t; taken together constraints (2.85-86) prevent a node from being visited more than once. Constraints of type (2.87) ensure that only one arc can be used for each sequence t. Curtin and Biba (2011) utilize the Vajda based framework above to formulate the TRANSMax model as follows:

i, j = indices of nodes that comprise the network
r = the index of arcs comprising a route
m = the number of nodes in the network
R = the maximum number of arcs within a route
A_{ij} = the service value associated with the arc from i to j
M_i = the service value associated with node i
d_{ij} = the length of the arc from node i to node j

D = the maximum length of the route

x_{ijr} = the decision value = to 1 if an arc from *i* to *j* is chosen in step *r* and 0 otherwise

$$Maximize \quad Z = \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{r=1}^{R}(A_{ij} + M_i)x_{ijr} \tag{2.88}$$

Subject to

$$\sum_{i=1}^{m}\sum_{r=1}^{R} x_{ijr} \le 1 \;\forall\; j = 1,2,\ldots,m \tag{2.89}$$

$$\sum_{j=1}^{m}\sum_{r=1}^{R} x_{ijr} \le 1 \;\forall\; i = 1,2,\ldots,m \tag{2.90}$$

$$\sum_{i=1}^{m} x_{ijr} - \sum_{i=1}^{m} x_{ji(r+1)} = 0 \;\forall\; j = 1,2,\ldots,m;\; r = 1,2,\ldots,R-1 \tag{2.91}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m} x_{ijr} = 1 \;\forall\; r = 1,2,\ldots,R \tag{2.92}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{r=1}^{R} d_{ij}x_{ijr} \le D \tag{2.93}$$

$$x_{ijr} \in \{0,1\} \;\forall\; (i, j, r) \tag{2.94}$$

where the objective (2.88) is to maximize the service values of the arcs, A_{ij}, as well as the service values for the nodes, M_i, using a total of *R* arcs. Constraints of type (2.89) require that for each node *j*, at most one arc across all sequences used on the path is allowed to be used to enter that node. Constraints of type (2.90) require that for each node *i*, at most one arc across all sequences may be used to leave that node. In effect these constraints, taken together, ensure that a path will only be able to enter and leave a node once. Constraints of type (2.91) require that there be a sequentially connected path. In this case, for each node *j* such that *j* is not on the first or last sequence, if an arc in sequence *r* enters node *j* then the *r*-th plus one arc in the sequence must leave node *j*. This constraint type will ensure that there is a connected path; the first and last sequence numbers are excluded as those sequences represent travel from and to the origin and

destination nodes respectively. Constraints of type (2.92) ensure that there will be one arc chosen for each sequence *r* on the route. It should be noted that by specifying a total of *R* sequences, the optimal path may not be found as it could be forced to include more sequences than required in the optimal solution. Additionally, there may not be enough sequences specified which are required to obtain a truly optimal solution. Constraint (2.93) requires that the route must not have a distance/cost larger than a prespecified distance/cost *D*. Since there is no objective function that minimizes overall path length, this constraint is used to ensure that a route does not have a cost/distance which exceeds a certain cost/length. Curtin and Biba chose to model transit in this manner as they note that many covering-path models do not optimize in terms of a user perspective but from an operational perspective (i.e. the models are oriented to find a least cost covering path as opposed to a path which optimizes the coverage of particular areas). By including a maximum cost/distance, Curtin and Biba can ensure that a user would experience a level of service which in the maximum case would be *D* cost/length. In essence, their goal is to ensure a user would not spend say an hour on a bus line, but rather a maximum of 30 minutes. This model framework could, however, be modified so that the maximum path cost/length given in constraint (2.94) is removed and an objective added which minimizes overall length. Constraints of type (2.95) are binary constraints used to ensure that there are no fractional solutions to the problem.

It is also important to note that the two constraint types (2.89 and 2.90) taken together could *a priori* ensure that a truly optimal solution is never found as the assumption built into this model, just as in other covering path models, is that an optimal solution will never cross over itself (generating a loop) or involve an attached loop as an

optimal covering strategy. By restricting a node to having one entering and one exiting arc, this will ensure that a node is never visited more than once. As was proven in Niblett and Church (2016), the assumption that a route will never involve a loop in an optimal solution is invalid and thus this model will also suffer from this limitation.

Curtin and Biba (2011) also note that their model is flexible and can be modified to fit various needs based upon particular planning cases. For example, they recognize that a salesman tour style route may be desired in a route planning context; as was noted in the introduction, this is the case for a vast number of medium sized cities across the US. That is, it may be desirable to find a route which begins and ends at a particular place – i.e. a transit center. This also would seem to address the assumption that a route can be traversed in either direction; however, as was noted for constraint types (2.89-90), the formulation does not allow a node to be revisited and thus the efficacy of this type of constraint in modeling "real world" routes is questionable. Nevertheless, Curtin and Biba give their generic loop constraint (2.95) as:

$$\sum_{j=1}^{m} x_{ij1} - \sum_{j=1}^{m} x_{ijR} = 0 \quad \forall\, i = 1,2,\ldots,m \qquad \textbf{(2.95)}$$

In effect the addition of this constraint type would require that for each node *i* either an arc leaving the node on sequence one and an arc entering the node on the final sequence *R* of the route will be chosen or node *i* will not be the starting and ending point of the route. If it is not chosen, node *i* could still be included on some other sequence for the route. However, this constraint only forces the route to return to a node at which it originates. In effect it creates a singular tour; since constraint (2.89) and (2.90) do not allow any node (including the origin and destination node) to be entered twice; this would

prevent any attached loops. Thus, the form of TRANSMax utilizing constraints of type (2.95) would be more akin to that of a covering TSP.

Curtin and Biba also note that it may be desirable to have a route which originates from and ends at a particular point of interest, such as a transit center. This form of the model is similar to the form described above; the difference is that instead of trying to determine the best possible tour given *R* sequences over the entire network, we now wish to find the tour which is guaranteed to originate and end with a specific location. To satisfy this type of need, Curtin and Biba proposed adding constraints (2.96) and (2.97):

$$\sum_{j=1}^{m} x_{sj1} = 1 \qquad \textbf{(2.96)}$$

$$\sum_{i=1}^{m} x_{isR} = 1 \qquad \textbf{(2.97)}$$

where *s* represents the point of interest that must be the terminus. Constraint (2.96) requires that an arc must be taken to leave the point of interest and constraint (2.97) requires that the last arc used on the route must be used to enter the point of interest. In effect constraints (2.96) and (2.97) will force the route to begin and end at a prespecified node rather than determining the optimal tour given *R* total arcs for a route. Curtin and Biba also note that there are two other variations which can be used to find an optimal route given *R* sequences. The first case defines where one may wish to require the route to end at a given terminus node while having no specific origin node. This case can be defined through constraint (2.98) where *e* represents the desired terminus node. The second case is similar, but instead of specifying a required terminus node

$$\sum_{i=1}^{m} x_{ieR} = 1 \qquad \textbf{(2.98)}$$

and no specific origin, the requirements are inverted wherein we now must depart from a specified origin node with no specific terminus node defined. This case is given by constraint

$$\sum_{j=1}^{m} x_{ej1} = 1 \qquad \textbf{(2.99)}$$

(2.99). Where *e* now represents the specified origin and *r* equal to one represents the first sequence to be used. Curtin and Biba note that all of the basic forms above can be further modified to accommodate areas that may have a significant service demand area such as business parks or other large employment centers through the use of waypoints. In this case a waypoint is defined as a known point of interest/demand, but is not desirable as an origin or terminal node. Curtin and Biba's waypoint constraint can then be defined as given in (2.100):

$$\sum_{j=1}^{m}\sum_{r=1}^{R} x_{wjr} + \sum_{i=1}^{m}\sum_{r=1}^{R} x_{iwr} \geq 2 \qquad \textbf{(2.100)}$$

where *w* represents the waypoint node desired on the route. The first summation accounts for all arcs which leave the waypoint node, *w*, across all sequences. The second summation accounts for all arcs which enter waypoint *w* across all sequences. Taken together with constraints (2.89), (2.90), and (2.92), this constraint will ensure that the waypoint is visited exactly one time and is not a beginning or ending point of a route. However, even though the basic TRANSMax model can be modified to meet various forms which suit a set of particular requirements such as way points, starting and ending nodes, etc. the TRANSMax model is unable to use attached loops/tours. Although Curtin and Biba do propose a form which creates a tour (i.e. a route that begins and ends at the same point) which was described above, this form does not allow for attached or

embedded tours within the route. This is due to the fact that such cases are prevented *a priori* through the use of Vajda style constraints. In this case it is constraints (2.89) and (2.90) which force a node to be entered and exited exactly once which prohibits an attached loop from occurring. If you remove these constraints from the model, then the Vajda framework falls apart and it is likely you would be left with a series of sub-tours as a solution, which defeats the purpose of using a larger set of variables x_{ijr} in the first place – that is, building a model which prevents unattached cycles without the need of an iterative constraint addition and solution procedure.

As was noted above, the last set of papers we will examine is comprised of problems that are Multi-Path Covering problems. There are four modeling formulations that have particular relevance to the original work of John Current as well as fit into the transit route modeling research quite nicely. The first paper on multi-path models is that of Boffey and Narula (1998). Boffey and Narula were the first to formulate the multi-path covering problem. Although in their paper they limit the number of paths to 2, they note their formulation could be expanded to include any number of paths. The practical applications for their model are of course in transit planning where one may want to design multiple routes, but it could also have application in problems where one may require the use of a number of routes for the same product or service. These can be such things as deliveries (such as newspaper distribution or mail delivery), mobile services (such as mobile glass repair, plumbing), electricity transmission/distribution, or even tourism (sightseeing, etc.). Boffey and Narula's model uses the Maximum Population Shortest path problem as a base framework (See (2.36) in the Key Problems section above). It is also important to note here that Curtin and Biba use *r* to represent the

chosen sequence of arc selection of a route whereas Boffey and Narula use r to differentiate between several different routes. The formulation of the Multi-Path Covering Problem involves the following notation:

i, j, k = indices of nodes that comprise the network
a_k = the population at node k
r = the number of paths that one desires to model, in this case the formulation uses 2
O = the origin node
D = the destination node
V = the set of vertices comprising the graph
F_i = the set of nodes that are reached by an arc from node i
T_j = the set of nodes that are connected by an arc to node j
d_{ij} = the length of the arc from node i to node j
x_{ij}^r = 1 if an arc from i to j is chosen on path r and 0 otherwise
y_k^r = 1 if node k is covered by path r and 0 otherwise

$$Maximize \quad Z = \sum_{k \in V} a_k (y_k^1 + y_k^2) - \sum_i \sum_{j \in F_i} d_{ij} (x_{ij}^1 + x_{ij}^2) \qquad \textbf{(2.101)}$$

Subject to

$$\sum_{j \in F_O} x_{Oj}^r = 1 \; \forall \; r = 1, 2 \qquad \textbf{(2.102)}$$

$$\sum_{i \in T_D} x_{iD}^r = 1 \; \forall \; r = 1, 2 \qquad \textbf{(2.103)}$$

$$\sum_{i \in T_k} x_{ik}^r - \sum_{j \in F_k} x_{kj}^r = 0 \; \forall \; r = 1, 2; \; i, j \neq O, D \qquad \textbf{(2.104)}$$

$$\sum_{j \in T_i} x_{ji}^r - y_i^r = 0 \; \forall \; i \neq j; r = 1, 2 \qquad \textbf{(2.105)}$$

$$y_k^1 + y_k^2 \leq 1 \; \forall \; k \in V \qquad \textbf{(2.106)}$$

$$x_{ij}^r, y_k^r \in \{0,1\} \; \forall \; i, j, k, r \qquad \textbf{(2.107)}$$

The objective (2.101) of the Multi-Path Covering Problem is a multi-objective formulation that involves maximizing the population covered and minimizing overall path length for each of the two stated paths (i.e. r = 2). Constraint (2.102) requires that one arc be used to leave the origin node for each path r and constraint (2.103) requires that each path reach a terminus node, D. In this case each path begins and ends with the

same origin and destination nodes; however, these could be changed for each path *r* so that each path would have a unique starting and ending node. Constraints of type (2.104) represent balance constraints for each path *r*; for each intermediate node – that is, a node that is not the origin or destination node – which is entered by an arc on the path must then be exited by an arc on the path that leaves the intermediate node. Constraints of type (2.105) represent the coverage constraints for each path *r*. In this case, in order for node *i* to be covered by path *r*, path *r* must utilize an arc which enters node *i*. It should be noted that there are no coverage distances used in this framework as this formulation is based upon the Maximum Population Shortest Path formulation where a node is considered covered only if it is directly visited by the covering-path. If one wished to transform this formulation to a Maximum Covering Shortest Path style covering-path problem, one could set up a maximum service distance, *S*, and utilize a neighborhood set defining the set of all nodes *j* that are within the service distance of node *i* (i.e. $N_i = \{ j \mid d_{ij} \leq S \}$ in place of set T_i in (2.105). Constraints of type (2.106) require that only one path can cover node *k*. This will ensure that there is no double coverage and the two paths are spatially unique. If constraint type (2.106) was not added to the formulation, each path *r* would follow the exact same route as each path would simply follow the best route and count coverage twice. Constraints of type (2.107) represent binary conditions for the *x* and *y* variables used on each path to represent whether they have been used or not used in the solution.

Boffey and Narula also proposed a formulation where the value of covering a node more than once can be captured through the use of two new binary variables, u_k

and v_k, where $u_k = 1$ if node *k* is covered by one path and v_k represents if node *k* is covered by both paths. The objective function for coverage can then be modified to type (2.108)

$$\sum_k a_k (u_k + \varepsilon v_k) \qquad \textbf{(2.108)}$$

where ε can range from a value of zero through one depending on the emphasis one wishes to place on the benefit/value of secondary coverage. A value of zero would represent no additional coverage benefit while a value of one would represent a benefit equal to that of covering the node once. Constraints of type (2.106) are then modified to the form in (2.109) and additional constraints of type (2.110) are added to the formulation.

$$y_k^1 + y_k^2 = u_k + v_k \quad \forall\, k \qquad \textbf{(2.109)}$$

$$v_k \le u_k \quad \forall\, k \qquad \textbf{(2.110)}$$

A constraint of type (2.109) will ensure that if one path travels to node *k*, u_k is allowed to be one since the coverage variables are binary. However, in order to prevent u_k from being zero in value while v_k is positive, constraints of type (2.110) ensure that in order for v_k to be used in the solution, u_k must already have a value – this of course indicates that one of the paths has already visited node *k*. It should be noted that the assumption used in creating this constraint is that the associated benefit multiplier value ε will be the same at all locations at all times which may or may not be true.

The Multi-Path Covering formulation is important in that it represented the first real step towards creating a modeling framework which represents the "real world"

through the use of multiple covering paths. What is interesting is that Boffey and Narula note that spurs, or branch lines – i.e. loops – may be a very acceptable alternative with respect to covering. And yet, if one employs this model, the issue of sub-tour elimination comes into play. Just as in the MCSP and SCP it is highly likely that sub-tours will form when the problem is first solved. Thus, an iterative process such as the Dantzig, et. al. iterative process must be used to eliminate all disconnected tours or flow constraints such as Gavish and Graves (1978) that keep the route connected. In this case, it is very possible that the optimal solution would be prevented from being determined as Boffey and Narula use the Dantzig, et. al. sub-tour elimination process whose flaws were detailed at the beginning of this chapter.

An additional issue that the Multi-Path Covering formulation faces is that it does not account for bi-directional travel; that is, Boffey and Narula still assume travel from a prespecified origin to a prespecified destination for each path can simply be reversed to form a transit route. In other words, although the use of two bus routes could be modeled using this formulation, the model itself does not consider the fact that a bus may or may not travel in the opposite direction along that same route. Another transportation related issue which is related to the issue of directional travel is that the formulation does not consider the nature of the stop. For example, if one assumes a route can travel in both directions, it may be that an intersection may just be a four way stop where a person can easily cross the street and board the bus in the opposite direction. However, if one must cross a 4 or more lane expressway this may mean that it is not practical to say that a stop can, in fact, cover the same population on each side of the street. In short, although the

Multi-Path Covering Problem is a step forward, it still has a number of shortcomings that could be addressed; particularly if the model is applied in a transportation context.

We next turn this discussion to models that primarily focus on transportation design and expansion. Wan and Lo (2003) were among the first to formulate a transit specific design model. Their model minimizes the overall cost of a transit network while meeting transit demand. The objectives of their model are to simultaneously locate routes as well as handle the flow of passengers along these routes. The key assumptions of their model are as follows: 1) all origin and destination nodes must be covered; 2) there exists a route or a collection of routes with spare capacity to meet every OD demand; 3) every route is acyclic (i.e. a route does not contain, nor is it a part of, a tour); and 4) every route serves both directions of the same line with the same frequency. These assumptions are very standard in the literature but they do come with drawbacks. For example, the assumption that a route will be acyclic has been proven to be flawed and the assumption that a route will travel in both directions based on one path can also be problematic. Wan and Lo's formulation does attempt to address the issue of how one can best optimize multiple routes to better serve travelers. Wan and Lo utilize the following notation:

A = the set of arcs in a graph G
$a \in A$ = an arc from node i to node j
C = the homogenous transit capacity
c_a = the link cost for using an arc
R = The set of transit routes
r = index of routes
Ω = the subset of all acyclic routes within R
r_k = an individual route, k, that is within the set Ω
f_r = the frequency for each route r, r = min being the minimum and r = max being the maximum desired frequencies (e.g. headways on a transit line).

s, t = indices for the starting and terminus node pairs in the set W

q_w = the demand at w

q_{st} = the demand associated with travel from s to t

q_{st}^r = the demand associated with travel from s to t that is served by route r

d_{ij}^r = 1 if a route travels from node i to node j or 0 otherwise

$d_{su}^r d_{ut}^r$ = indicates a stop is made at u on route r between OD pair (s,t)

x_a^r = 1 if route r traverses arc a and 0 otherwise

M = a very large number

$$Minimize \quad Z = \sum_{r=1}^{R_{\max}} \sum_{a \in A} f_r c_a x_a^r \tag{2.111}$$

Subject to

$$\sum_{r=1}^{R_{\max}} q_{st}^r = q_{st} \ \forall (s,t) \in W \tag{2.112}$$

$$Cf_r - \sum_{(s,t) \in W} d_{su}^r d_{ut}^r q_{st}^r \geq \sum_{(u,v) \in W} d_{uv}^r q_{uv}^r \ \forall u \in W^O \tag{2.113}$$

$$Cf_r - \sum_{(s,t) \in W} d_{tu}^r d_{us}^r q_{st}^r \geq \sum_{(u,v) \in W} d_{vu}^r q_{uv}^r \ \forall u \in W^O \tag{2.114}$$

$$r_k \in \Omega, \quad \forall k = 1, \ldots, R_{\max} \tag{2.115}$$

$$f_{\min} \leq f_r \leq f_{\max} \tag{2.116}$$

$$0 \leq q_{st}^r \leq M(d_{st}^r + d_{ts}^r), \quad \forall (s,t) \in W, r = 1, \ldots, R_{\max} \tag{2.117}$$

$$x_a^r \in \{0,1\}, \quad \forall r = 1, \ldots, R_{\max}; a \in A \tag{2.118}$$

$$d_{st}^r \in \{0,1\}, \quad \forall s,t \in W^O \cup W^D \tag{2.119}$$

In this case, the objective function (2.111) is to minimize the operating cost of all transit lines where f_r represents how often route r is used (representing route headways), c_a represents the link cost for an arc, and x_a^r represents whether arc a is included in the solution for route r. Constraints of type (2.112) decompose each OD demand into route specific demands. Constraints of type (2.113) represent capacity requirements for each route r as it travels through stops on the route. In this case the authors assume that capacity is homogenous for the entire system and that every route or collection of routes

is able to meet all demands for a given route frequency. Thus, (2.113) represents the fact that there must be a route or series of routes that has capacity to serve demand.

$$\sum_{(s,t)\in W} d_{tu}^{r} d_{us}^{r} q_{st}^{r} \qquad \textbf{(2.120)}$$

The sum represented in (2.120) represents the passenger volume that is already aboard

$$\sum_{(u,v)\in W} d_{uv}^{r} q_{uv}^{r} \qquad \textbf{(2.121)}$$

route *r* at node *u* while (2.121) represents the boarding demand from node *u* onward. Constraints of type (2.113) handle travel volume in a forward direction and constraints of type (2.114) handle travel volume to be met in the opposite direction. It should be noted that although this would account for travel in opposite directions it does not allow for any loops to be used as the model is designed such that a route is only determined in one direction (i.e. all routes are by definition acyclic). These constraints are based on the inherent assumption that it would be possible to travel in the opposite direction without taking into account the true spatial layout of a route. Constraints of type (2.115) are used to represent that each route will be part of an acyclic set. In essence this set represents the set of all possible acyclic routes. Wan and Lo's applied formulation must use some form of tour elimination constraints in order to ensure feasibility and connectedness of identified routes. The easiest way to do this is the flow balance and sub-tour elimination constraints of Current et. al. (1985). Constraint type (2.116) sets limits for the lower and upper bound on line frequency for each route. This constraint attempts to address how often a route should be utilized. This is akin to how often a bus should run on the line. Constraint set (2.117) is used to allow positive demand on route *r* only when a route is traversed along an associated arc in either a forward or backward direction. In this case,

since demands may not be uniform, a large value *M* is used to make the number sufficiently large so as to accommodate possible demand but still ensure demand cannot be allowed if either direction variable $d_{st}^r + d_{ts}^r = 0$ as $M * 0 = 0$. Constraint types (2.118) and (2.119) are binary constraints on arc selection and direction respectively. If an arc is utilized or a route travels in a specific direction then these variables will equal one, zero otherwise.

As noted above, however, the set Ω cannot be known *a priori* and thus Wan and Lo must specify a method for preventing sub-tours and looped routes (i.e. routes that comprise, or even utilize, a loop). Although one could conceivably adapt a Dantzig, et. al. iterative tour elimination process, Wan and Lo do so through a mechanism inspired by Miller, Tucker, and Zemlin (1960), and their approach adds considerable computational complexity to an already complex problem. In order to explain their approach it is necessary to describe the variables they utilize. These are given below:

ψ_k^r = the sequence number for node *k* will be the *n*-th number of the sequence for route *r* or 0 otherwise

ξ_k^r = 1 if route *r* starts at node *k* or 0 otherwise

φ_k^r = 1 if route *r* ends at node *k* or 0 otherwise

$x_{hk}^r = x_a^r$ the authors use these variables interchangeably to represent an arc. In this case the *h* represents the node that forms the head of the arc and *k* represents the node at the tail.

N_k^- = the set of tail nodes comprising an arc entering node *k*

N_k^+ = the set of head nodes comprising an arc leaving node *k*

The expanded form of constraint type (2.115) which represents the Ω set is then given as:

$$\xi_k^r + \sum_{h \in N_k^-} x_{hk}^r \leq \psi_k^r \quad \forall\, k \in N \qquad \textbf{(2.122)}$$

$$\xi_k^r + M \sum_{h \in N_k^-} x_{hk}^r \geq \psi_k^r \quad \forall\, k \in N \qquad \textbf{(2.123)}$$

$$\psi_k^r \le M\left(\sum_{h\in N_k^-} x_{hk}^r + \sum_{h\in N_k^+} x_{kh}^r\right) \quad \forall\, k \in N \tag{2.124}$$

$$x_{hk}^r + \psi_h^r - M\left(1 - x_{hk}^r\right) \le \psi_k^r \quad \forall\, k \in N;\, h \in N_k^- \tag{2.125}$$

$$x_{hk}^r + \psi_h^r + \xi_k^r + M\left(1 - x_{hk}^r - \xi_k^r\right) \ge \psi_k^r \quad \forall\, k \in N;\, h \in N_k^- \tag{2.126}$$

$$\frac{1}{M}\sum_{a\in A} x_a^r \le \sum_{k\in N} \xi_k^r \tag{2.127}$$

$$1 - \frac{1}{M}\left(1 - \sum_{a\in A} x_a^r\right) \ge \sum_{k\in N} \xi_k^r \tag{2.128}$$

$$\sum_{k\in N} \varphi_k^r = \sum_{k\in N} \xi_k^r \tag{2.129}$$

$$\sum_{h\in N_k^-} x_{hk}^r \le 1 \quad \forall\, k \in N \tag{2.130}$$

$$\sum_{h\in N_k^+} x_{kh}^r \le 1 \quad \forall\, k \in N \tag{2.131}$$

$$\sum_{h\in N_k^-} x_{hk}^r + \xi_k^r - \varphi_k^r = \sum_{h\in N_k^+} x_{kh}^r \quad \forall\, k \in N \tag{2.132}$$

In this expanded set of constraints used to define the feasibility of any design routes, Wan and Lo use a node labeling approach to delineate attributes of the model, the variables for which are given above. In this case, constraints of type (2.122) and (2.123) represent a bound on the value of a sequence. These labeling constraints are inspired by the Miller, Tucker, and Zemlin TSP sub-tour prevention constraints developed for traveling salesman problems in that ψ represents the value of the sequence of the path; constraints of type (2.122) and (2.123) stipulate that in order for route *r* at sequence *k* to be labeled as the next sequence of the route, an arc must have entered node *k* on the previous sequence. Constraints of type (2.124) ensure that a node on a route can only have a sequence value if an arc enters or leaves that node. Constraints of type (2.125) and (2.126) ensure that the value assigned to each sequence – i.e. each ψ – along the route must be an increment of 1 larger than the sequence value of the node which entered *k*. Constraints of type (2.127) and (2.128) are meant to ensure that there is one starting node that exists while

constraints of type (2.129) ensure that there is also a terminus node for each route if an origin node for the route exists. Constraints of type (2.130) require that, for each node *k*, only one arc may enter node *k* and constraints of type (2.131) require that only one arc may leave node *k* on a specific route, *r*. Taken together, constraints of type (2.130) and (2.131) are used to prevent tours/cycles from occurring. Constraints of type (2.132) are essentially flow balance constraints where if a node is entered by an arc there must be a corresponding arc that leaves that node with the exception of the origin and destination nodes. In effect these additional constraints are needed to ensure that the route is a path rather than a tour, unless $\xi_k^r = \varphi_k^r$.

One of the biggest drawbacks of this model formulation is that as the size of a problem increases, the number of constraints and variables increase dramatically, particularly given the tour elimination constraints that Wan and Lo add to the model. This is a hard problem that is difficult to solve to optimality. The second major issue is that Wan and Lo note that their formulation is not a truly linear model as the objective and several of the introduced constraints are not linear – e.g. $f_r * x_a^r$ in the objective (2.111). Wan and Lo do attempt to linearize some of these constraints, but again, doing so adds more computational overhead to the formulation. The last issue is that their model explicitly prevents tours – whether attached or embedded – from occurring. Thus, it is very possible a model of this type could exclude a truly optimal solution. However, the formulation itself does attempt to address several crucial issues such as a user versus operator perspective in terms of quality of service versus cost, and is the first true attempt to do so in the context of a covering path framework. One should also note that enough

capacity is provided between each origin and destination to handle all traffic, but does not address how long users might have to wait for a bus that is not already at capacity.

Wu and Murray (2005) formulated the Multiple Route, Maximal Covering Shortest Path problem to optimize transit quality and system access. The authors note that transit access – that is, the ability of a person to enter and ride on a transit system – is of utmost importance in urban regions. In particular, emphasis is placed on maximizing the spatial extent of system access in order to ensure that those with mobility problems (i.e. senior citizens, persons with a disability) and those at the lower end of the socio-economic ladder are able to adequately move about the city in pursuit of better jobs, schooling, etc. An equally important objective is that of service quality. This includes things such as: convenience, travel time, comfort, information access, reliability, safety, etc. (Levinson, 1992). The use of interactive trip planning tools can also enhance a level of service such as the Google Maps' transit planner (Huang and Peng, 2002; Peng and Huang, 2000). Improvement in reliability has been sought through the use of automatic vehicle location data associated with bus arrival or departure at specific stops (Cathey and Dailey, 2003; Yu et. al., 2011) and by implementing transit signal priority (Ling and Shalaby, 2003; Li et. al., 2011). Yet an underlying theme is travel-time performance, which is why travel time remains an important aspect of service quality, as noted by Newman and Kenworthy (1999). The name of the model of Wu and Murray (2005) is a bit misleading as it does not design routes nor does it seek to find maximal covering shortest paths. It actually selects, among existing bus stops, a subset of stops that, when kept, improve service times while keeping access coverage as high as possible. Wu and Murray's model notation and formulation are given below as:

i, j, k = indices of existing bus stops
r = index of existing transit routes
m = index of ridership service areas
o = index of origin terminals for transit routes
d = index of destination terminals for transit routes
l_{ij} = the transit travel distance from stop i to stop j
v_{ij} = the cruise speed from stop i to stop j
δ_i = the total delay time at stop i associated with bus acceleration, deceleration, and door opening and closing
t_{ij} = the total travel time between stop i and stop j without intermediate stops
a_m = the potential ridership demand in service area m
Δ_{mj} = the shortest travel time/distance from service area m to stop j
S = the suitable access service standard
$N_m = \{ j \mid \Delta_{mj} \le S \}$ the set of stops that can service area m within the access standard
y_{rm} = 1 if service area m is covered by route r and 0 otherwise
z_{odij} = 1 if a directed arc from stop i to stop j is included in path from o to d and 0 otherwise
x_j = 1 if stop j is selected to remain in the system and 0 otherwise

$$Maximize \quad Z = \sum_m \sum_r a_m y_{rm} - \sum_o \sum_d \sum_i \sum_j t_{ij} z_{odij} \tag{2.133}$$

Subject to

$$\sum_{j \in N_m} x_j \ge y_{rm} \quad \forall r, m \tag{2.134}$$

$$\sum_j x_{odoj} = 1 \quad \forall o \in O, d \in D \tag{2.135}$$

$$\sum_i x_{odid} = 1 \quad \forall o \in O, d \in D \tag{2.136}$$

$$\sum_i z_{odij} - \sum_k z_{odjk} = 0 \quad \forall o \in O, d \in D, j \in J, j \notin O, j \notin D \tag{2.137}$$

$$\sum_i z_{odij} = x_j \quad \forall o \in O, d \in D, j \in J \tag{2.138}$$

$$x_i, z_{odij}, y_{rm} \in \{0,1\} \quad \forall i, j, o, d, r, m \tag{2.139}$$

In this transit stop selection model, the objective (2.133) is to maximize ridership while minimizing total system travel time. The first objective term maximizes the potential ridership in each service area that is covered by a located bus stop within a suitable access time/distance. The second objective term minimizes the total bus travel

time between terminal origin-destination pairs for all routes – in other words the overall operation time for each route in a transit system is minimized. Constraints of type (2.134) are coverage constraints for each route and service area. If a stop is located on a route within the access standard of service area *m* then that area is considered covered. Constraints of type (2.135) and (2.136) ensure that an arc is selected to leave an origin terminus and an arc is selected to arrive at a destination terminus for each transit route. Constraints of type (2.137) are flow balance constraints which ensure that if an arc on a route enters a node, then that arc must subsequently leave that node with the exception of all origin and destination nodes for each route. Constraints of type (2.138) are used to track where bus stops are sited. In this case, a bus stop cannot be located on a route unless it is connected by an arc on the route from *o* to *d*. Constraints of type (2.139) are binary constraints which ensure that stops are either assigned or not assigned, arcs cannot be partially used, and coverage is not partially assigned.

Wu and Murray (2005) also give a metric that allows one to calculate the value for t_{ij} as a function of travel time along a direct route or as a function of a transfer between two or more routes. The formula for capturing this value is given in (2.140):

$$t_{ij} = \begin{cases} \frac{1}{2}\delta_i + \frac{l_{ij}}{v_{ij}} + \frac{1}{2}\delta_j, & \text{if stops } i \text{ and } j \text{ are on the same route} \\ \infty, & \text{if stops } i \text{ and } j \text{ are on different routes and not transfer stops} \\ T_{ij}, & \text{if stops i and j are transfer stops on different routes} \end{cases}$$

(2.140)

where δ represents the total delay time due to acceleration/deceleration and door opening and closing, l_{ij} represents the transit travel distance from stop *i* to stop *j* and v_{ij}

represents the cruising speed from stop *i* to stop *j*. Thus, if stop *i* and stop *j* are on the same route, the estimated travel time between these stops will be half the delay at stop *i* (representing the time it takes for passengers to board and the bus to leave) plus the time it takes to go from stop *i* to stop *j* plus half the delay at stop *j* (the time it takes for the bus to stop and passengers to get off). If *i* and *j* are transfer stops on different routes, then T_{ij} represents the amount of time it takes to transfer from one route to the next (in essence, the amount of delay). If stop *i* and *j* are not on the same route or a transfer stop, then their value is infinite as it is impossible to transfer or continue on to stop *j*. Wu and Murray also note that each objective can be weighted in a similar manner as was applied in the MCSP which would allow for the generation of a tradeoff curve. In this case, such a curve would represent the importance of covering each service area with the importance of efficient service.

By the fact that routes are held and fixed and stops are being localized, the model of Wu and Murray will not be affected by the possibility of sub-tours as no routing variables are included. However, by restricting the design in this fashion it is possible that an optimal routing and stopping scheme will be overlooked. In fact, if one expected to see elements of real world routes as well as the fact that it has been proven that attached tours are an effective covering strategy (Niblett and Church, 2016), one would expect that there would be at least one attached tour in many systems. Yet in this case, due to the restriction on the routes and network, what will truly be optimized is the siting of bus stops such that as much of the access area can be covered as possible using fewer stops. If one were to open this model to changes in route alignment by allowing routing to occur on a true road network, it is certain that a model of this nature would include

sub-tours in an initial solution and thus would need a sub-tour elimination process to account for their occurrences. Overall, the requirement to follow a pre-set route alignment is by far the biggest drawback of this model.

Another limitation of this model is that stops are assumed to provide coverage regardless to the direction the bus travels. This means that the problem may not be truly captured as they have formulated it. A related issue lies in the fact that it is assumed that a bus will be able to reverse direction and begin to travel in the opposite direction of the stated covering-path at an origin or destination terminal. Although many cities do employ "transit centers" this assumption may not hold across the board – for example a bus may actually continue in the same direction and instead simply change the route number that it is traveling on. An example of such a case is that involving Line 6 and 11 in the Santa Barbara Metropolitan Transit District. The strength of their formulation is that it was one of the first attempts to improve service and route design with respect to access and efficiency, and it did so with a formulation much more concise than that of Wan and Lo. However, to reiterate what was mentioned above, one of the major drawbacks of this formulation is that it is limited in terms of network scope. That is, the authors constrained the model to select from pre-existing stops while keeping routes fixed. The last paper that will be discussed in this section attempts to address some of these issues while extending existing routes.

Matisziw et. al. (2006) propose a formulation for strategic route extension for new urban areas. This formulation is referred to as the Maximal Covering Route Extension Problem (MCREP). The formulation is designed such that transit agencies can add to their existing routes in order to minimize user disruption and dissatisfaction as well as

cover new urban areas as efficiently as possible within a standard of access. In particular, the authors have formulated their model such that new areas will be served via a pre-specified number of potential new stops. The MCREP is formulated as follows:

i, j = index of potential candidate stops
k = index of demand areas
r = index of clique sets (Entire set = R)
T = the set of beginning terminal nodes t
E = the set of destination (dummy) nodes e
$N_j = \{\text{node } i \mid arc(i, j) \text{ is defined}\}$
$N_t = \{\text{node } t \mid arc(t, j) \text{ is defined}\}$
$N_e = \{\text{node } e \mid arc(i, e) \text{ is defined}\}$
a_k = the potential demand in area k
d_{ij} = the distance between stop i and stop j when i and j are directly connected
p = the number of stops to locate
M_k = the set of stops j which cover demand k
C_r = the subset of stops for which one j covers the same demand as other stops
z_j = 1 if potential stop j is used as a stop and 0 otherwise
y_k = 1 if demand k is covered and 0 otherwise
x_{ij} = 1 if arc $\{i, j\}$ is on the solution path and 0 otherwise

$$Maximize \quad Z = \sum_k a_k y_k - \sum_i \sum_j d_{ij} x_{ij} \tag{2.141}$$

Subject to

$$\sum_{j \in N_t} x_{tj} = z_t \;\; \forall \, t \in T \tag{2.142}$$

$$\sum_{i \in N_e} x_{ie} = z_e \;\; \forall \, t \in T \tag{2.143}$$

$$\sum_{i \in N_j, i \neq e, j} x_{ij} - \sum_{l \in N_j, l \neq t, j} x_{jl} = 0 \quad \forall \; j, \; j \neq t, e \tag{2.144}$$

$$\sum_{j \neq t, e} z_j = p \tag{2.145}$$

$$z_j - \sum_{i \neq j} x_{ij} \leq 0 \quad \forall \; j, j \neq t \tag{2.146}$$

$$z_i - \sum_{j \neq i} x_{ij} \leq 0 \quad \forall \, i, i \neq e \tag{2.147}$$

$$y_k \leq \sum_{j \in M_k} z_j \quad \forall \, k \tag{2.148}$$

$$\sum_{j \in C_r} z_j \leq 1 \quad \forall\, r \in R \qquad \textbf{(2.149)}$$

$$x_{ij}, y_k, z_j \in \{0,1\} \quad \forall\, i, j, k \qquad \textbf{(2.150)}$$

The objective (2.141) maximizes coverage and minimizes added route cost/length. Specifically, the first portion of the objective statement maximizes the demands that are covered while the second portion of the objective statement minimizes the overall cost/length of the route. Constraints of type (2.142) ensure that an origin node for the extended route, *t*, is part of an existing route. In this case, because we wish to extend routes into a new service area, the origin for the extended route could be located anywhere where a current route currently serves an area. Constraints of type (2.143) ensure that a route will end at an ending node, *e*, which lies on an existing route. Constraints of type (2.144) are balance constraints which ensure that if an intermediate node *j* is entered by an arc, there must also be a corresponding arc which leaves intermediate node *j*. This is done for all nodes with the exception of the origin and destination nodes for the extended route – that is all nodes except nodes *t* and *e*. Constraint (2.145) specifies that *p* stops along the route will be used. Constraints of type (2.146) ensure that if a stop is located at node *j* then there must be a corresponding arc which enters node *j* with the exception of the route extension origin node *t*. Constraints of type (2.147) ensure that in order for a stop to be located at node *i* there must be a corresponding arc which leaves node *i* with the exception of the destination node *e* for the extended route. Constraints of type (2.148) ensure that in order for node *k* to be considered covered, there must be a stop located at a node that is able to cover *k*. Constraints of type (2.149) are clique constraints that will prevent redundant coverage. That is, only one stop is allowed to be selected from among a set of stops that provide the

same spatial coverage. Constraints of type (2.150) are binary constraints which stipulate which variables must be either one or zero in value.

Just as in the SCP/MCSP derived covering-path problems, the potential for sub-tours exists in this model formulation. It is very likely that sub-tours would appear, particularly on a large graph. In this case, the authors suggest a sub-tour elimination procedure similar to that used by Current et. al. which is based on Dantzig, Fulkerson, and Johnson TSP sub-tour elimination constraints. Thus, in order to obtain a solution which does not include tours, a model must be solved, sub-tours identified, constraints need to be added, and then the problem can be re-solved and the process repeated if the need arises. As noted previously in this section, the use of Dantzig, Fulkerson, and Johnson style constraints used by Current et. al. and most of the other covering-path formulations have the potential to exclude truly optimal solutions from being determined. Another potential problem is that these routes are assumed to be bi-directional; so even though a path may be found, it is possible that a bus may not be able to make a U-turn, or, in the worst case scenario, a bus may not physically be able to travel in the opposite direction due to the fact that the route may utilize a way one street. It should also be noted that an emphasis on keeping the alignment of current routes static – i.e. relatively unchanged – could adversely impact the model. Although one may not want an established transit line to shift its route alignment too much in order to avoid angering the current ridership, it is possible that a marginal change in the route alignment, or even a system wide re-alignment, could substantially improve service in terms of access, accessibility, and system cost.

Nevertheless, the Maximal Covering Route Extension Problem is easily formulated and solved on large networks using current computational solvers which is in contrast to the Multiple Route Transit Network Design problem formulated by Wan and Lo. Matisziw et. al. applied the MCREP to newly developed areas in Franklin County, Ohio which includes the Columbus, Ohio metropolitan area. What is interesting about their results is that one of their proposed extensions includes a nearly attached tour at two points along the route as seen in Figure 2.6 below. Thus, if an EAST constraint process were used, it is possible that more such loops could appear in a solution. Therefore, the task is not only to determine a method of accounting for transit routes in both directions, but also how routes can utilize loops to efficiently extend coverage.

Up to this point we have reviewed the core set of covering-path models and formulations as well as extended models which deal with covering-path applications such as public transportation. We have seen that these models utilize a framework which has been proven to be flawed and we have seen that they do not entirely capture the problem based upon the assumptions that a path can travel in both directions. It is also assumed that busses can easily change direction at the terminal points of the route – that is their

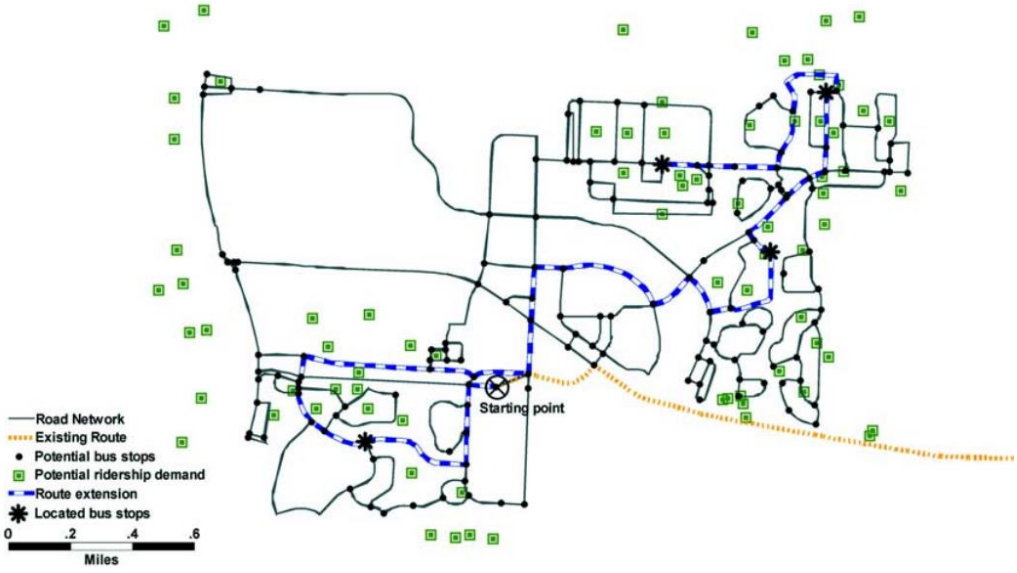


Figure 2.6 - Example of a nearly attached tour (Matisziw et. al., 2006)

origin and destination points – for travel in the opposite direction. Moreover we have seen that these models don't easily capture the advantage that a route could provide through the utilization of an attached or embedded loop and are in fact often encouraged *not* to utilize such features either through explicit prevention such as in Vajda derived models or through tour-elimination processes such as those modified from Dantzig, Fulkerson, and Johnson. What we have yet to cover is the area of heuristics – or methods used to find a reasonably 'good' solution – which have been developed to solve these models. Therefore, it is pertinent to visit the heuristic work that has been developed to solve these problems in the next section.

2.4 Solution Procedures – Algorithms

This section will examine several key papers that have developed algorithms or heuristics used to solve covering-path problems. In order to provide a precise definition of terms, this dissertation defines an algorithm as a method or process that is used to obtain a guaranteed optimal solution to a problem while a heuristic is a process or method

which is used to obtain good if not optimal solutions to a defined problem but with no guarantee of optimality or no guarantee of how close to optimal such a solution might be. This is an important distinction as there are several algorithms which yield optimal solutions for specific problems – e.g. Dijkstra's Algorithm (1959) for shortest paths – while heuristics are often used to solve NP-Hard problems such as the p-Median problem – e.g. Teitz and Bart (1968). Often heuristics can produce optimal results; the key point is that there is no guarantee. Thus, when discussing methods for finding solutions to a stated problem we refer to an algorithm as a process which yields a verifiable optimal result and we refer to a heuristic as one which yields, hopefully a good, but not necessarily optimal, result.

This particular section will present several methods which have been used to solve covering-path type problems. This section will also include a brief discussion of shortest path algorithms, as covering path problems are a more complex form of a shortest path problem. It is important to note that there are many different heuristic approaches that could be applied – e.g. greedy, genetic, etc. – but it is imperative that the heuristic is an appropriate fit for the problem. A good heuristic for one problem type – e.g. Teitz and Bart for the *p*-Median – does not necessarily mean it is an appropriate fit for covering-path type problems.

One of the classic algorithms for determining shortest paths through a network is Dijkstra's Algorithm (1959). Although Dijkstra did not come up with the first shortest path algorithm – the Bellman-Ford-Moore algorithm was simultaneously developed between 1956 and 1958 (Ford, 1956; Bellman, 1958; Moore, 1959); Minty (1957) even

suggested using knots on a series of strings.[8] Schrijver (2012) gives an excellent overview of the development of methods used to solve shortest path problems. However, methods that have been applied in related covering-path problems are those based upon k^{th}-shortest paths. In general k^{th}-shortest path algorithms find the *k* paths connecting a given source/destination pair in order of total path length (Hoffman and Pavley, 1959). In other words, k^{th}-shortest path algorithms can find the 1^{st} shortest path, 2^{nd} shortest path, etc., out to the k^{th}-shortest path in order of lengths. This problem was first defined by Bock, et. al. (1957) although the first published paper – using a different approach – is by Hoffman and Pavley (1959). Interestingly, there are two different versions of this problem; the k^{th}-shortest path which does not include loops (i.e. tours) and the k^{th}-shortest path which does include loops (i.e. tours). The first efficient non-looping algorithm was developed by Yen (1971)[9] and the most efficient looping algorithm by Eppstein (1998)[10]. It is not surprising that a *k*-shortest path would have the possibility of looping through a node as we have seen that it can be a very efficient covering strategy in the covering-path formulation. In a shortest path context it makes sense that if one has a very short segment the path may loop around an arc forming a simple tour until another path exceeds the efficiency of doing so. Coutinho-Rodrigues et. al. (1999) developed a k^{th}-shortest path based algorithm with respect to finding unsupported non-dominated solutions for bi-objective shortest-path formulations (i.e. SCP/MCSP, etc.). However, with respect to maximal covering, shortest path problems there has been little work done with respect to

[8] If one develops a network using a series of strings in which nodes represent cities and string lengths distance between each city one can pull the knots representing the OD pair apart to find the shortest path. The tautest string or set of strings represents the shortest path between the OD pair.

[9] Although Yen's Algorithm was the first to have a known order of complexity, other approaches have been able to further improve computational time (Lawler, 1972; Perko, 1986; Martins and Pascoal, 2003)

[10] Although the paper was published in 1998 Eppstein first reported his algorithm at the 35^{th} Annual Symposium in the Foundations of Computer Science (IEEE) in 1994.

exact algorithms. The bi-objective nature of the problem makes the development of such algorithms particularly complicated as there can be considerable variation regarding emphasis that is placed on each objective. However, methods used to solve integer problems as well as determine the non-inferior frontier are quite useful and these are frequently solved through algorithmic methods. Therefore, the first algorithmic solution method I would like to discuss is a process used to find integer solutions to a problem which is often referred to as the 'Branch and Bound' method or algorithm. The second method described relates to finding the non-inferior solution frontier and is called the Non-Inferior Set Estimation (NISE) method.

In order to further explain the implications of modeling problems which require integer solutions, a brief set of definitions will be given. When formulating a linear programming problem/model, a graph where solutions are plotted as a function of variables can be defined. The area representing the values associated with the variables of the problem can be defined as a representation of the solution space. If we plot the line (or hyperplane) defined by each constraint, we can begin to define a bounded region which represents a limit on possible decision variable values. Depending on the definition of the constraint, the feasible solutions to the problem will lie either on, above, or below these constraint lines (or hyperplanes). Thus, these lines (or hyperplanes) then define the region in which we have a feasible solution or the region of feasibility. If we assume a simple problem where there exists a set of feasible solutions such as the problem given by (2.151-153):

$$Maximize \quad Z = cx \tag{2.151}$$

Subject to

$$Ax \leq b \tag{2.152}$$

$x \geq 0$ **(2.153)**

where x is an n by one vector, b is an m by one vector, and all other matrices have conformable dimensions, then a feasible region is then defined as the area which lies within the region defined by $Ax \leq b$ (2.152). Note, since we specify that the vector x is non-negative (i.e. greater than or equal to zero) in constraint (2.153) this will ensure that the region will fall in $\hat{A}$ space where all variables are greater than or equal to zero. The unique points at which one or more of these lines (or hyperplanes) intersect are defined as the extreme points of the problem which we simply refer to as extreme points. Therefore, extreme points represent a change in the limiting function for a particular feasible region. In other words, these points represent where the limiting factor switches from one constraint to another. In our sample problem the objective function (2.151) defined by cx will be maximized and thus the optimal solution for this problem will be one of the extreme points defined by (2.152) above.[11] When solving these linear models, one can use Dantzig's simplex method which pivots along these extreme points until the optimal solution is ultimately determined[12].

However, if we constrain a problem to be defined as integer or binary, the set of solutions within the feasible solution space is no longer a continuous region but is now defined to be a set of discrete points at which each point is a discrete integer-feasible solution to the problem. In effect this means that an extreme point which defines an optimal solution in continuous space may not represent an optimal solution for a discrete integer problem. Thus, a more computational intensive approach must be used to

[11] For a thorough explanation of mathematical programming I would refer one to Hillier and Lieberman's *Introduction to Operations Research*.

[12] There are possible nuances of cycling, which are often not a concern. However, they do present theoretical issues that need to be addressed when proving convergence.

determine the optimal discrete integer solution to the problem. There have been several methods proposed that will do this. One of the first to examine this issue was Gomory (1958, 1960) who suggested an algorithm in which additional constraints could be added to a problem. These added constraints are cutting planes which are used to reduce the feasible region in an attempt to prevent non-integer solutions from occurring at the edge of the region of feasibility containing the optimal all integer solutions. Although this is may be an effective way to solve the problem, it is not terribly efficient – especially as a problem grows in complexity, especially in terms of the number of integer variables involved.

Land and Doig (1960) developed what has since become known as the Branch and Bound Method/Algorithm. Rather than use a purely constraint-based approach to solve an integer problem such as Gomory; Land and Doig recognized that the relaxed integer problem – i.e. the relaxation of integer/binary constraints to non-negativity constraints – represents a bound to the integer problem. That is, a solution to the integer problem in the best case can never exceed that of the linear form of the problem. This is illustrated in Figure 2.7. Thus, Land and Doig recognized that a method could be devised wherein one can find the best bound and then 'branch' along the associated search tree until the solution matches the best bound found. The generic form can be stated as follows:

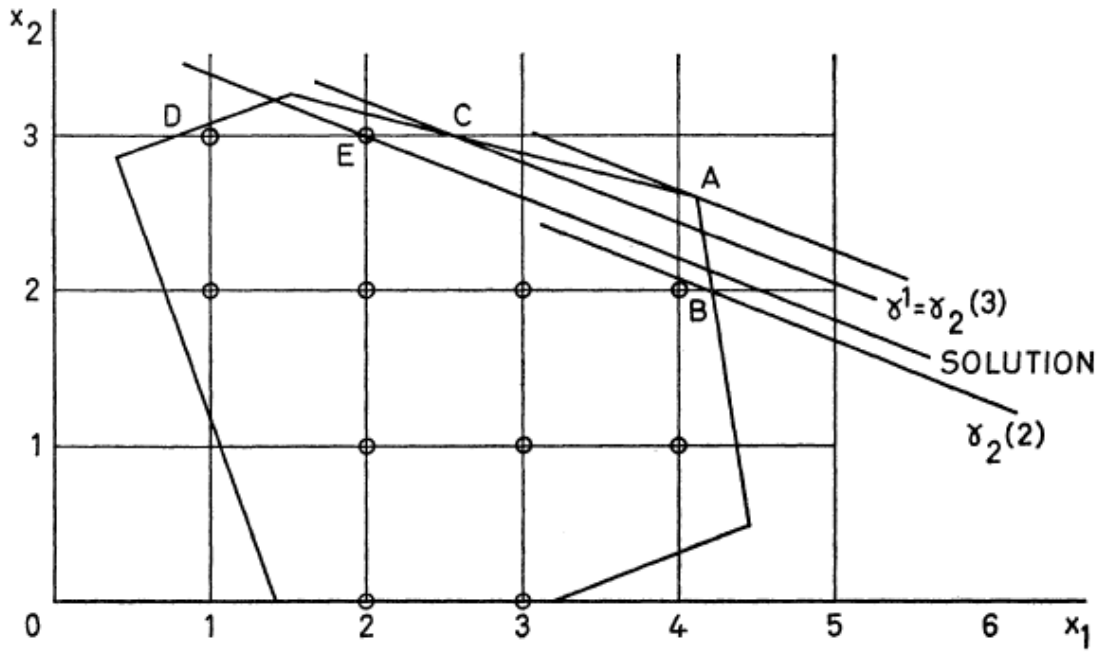


Figure 2.7 - Feasible Region to a linear problem and the associated integer solution given in Land and Doig (1960). On the graph Point A represents an optimal linear solution while Point E represents the optimal integer solution.

Step 0: determine a solution x_h to the problem (Note that this is a relaxed linear solution). Set *B* as the solution to x_h where *B* represents the best solution found by the linear relaxation.

Step 1: Determine all variables in the solution x_h which do not have integer values. These represent the variables on which we will 'branch.' In the case of the relaxed problem – if all variables are already integer stop, this is the optimal solution.

Step 2: Pick one of the 'branch' variable candidates and set it to be the associated floored integer value or the ceilinged integer value – e.g. if a variable has a value of 3.7 its floor is 3 and ceiling is 4. Determine the solution to the associated sub-problem using these values. If a solution to a sub-problem is not feasible then that branch can be removed from further exploration. If a sub-problem has an integer solution it can also be removed from the set of 'branches' that require further exploration as there will be no better solution than the current integer solution.

Step 3: Chose the next variable to 'branch' on which has an associated objective value closest to *B*. If all initial non-integer variables have been 'branched' set the best value found to be the best objective found associated with the branch variable that yields the best objective.

Step 4: Repeat steps 2-3 until an integer solution is found which equals the best objective or until all portions of the tree have been fathomed – i.e. no better solution can be determined based upon branching on that variable.

The nice thing about the branch and bound algorithm is that you can use solutions obtained from other heuristic methods to set bounds. If the best integer solution found by the branch and bound algorithm is found to be a solution determined by the heuristic, then you know that you have found the optimal solution. The branch and bound process is also relatively easy to code and there exist several methods which amend the basic branch and bound process to improve solution times.[13] In any case it is discussed here due to the fact that a form of the branch and bound algorithm is used by virtually all commercial solvers to determine optimal binary/integer solutions. Virtually all covering-path type problems are solved to optimality through the use of a branch and bound algorithm. Although the branch and bound algorithm will find an optimal solution to a problem given enough time, the practicality of doing so falls as problems increase in size.

Due to the bi-objective nature of covering path problems, it is essential to perform some kind of multi-objective analysis. One of the earlier methods developed with respect to multi-objective analysis, where the goal is to identify pareto optimal solutions, was that of Cohon et. al. (1979) who developed a method for identifying the non-inferior tradeoff curve. Cohon et. al. (1979) first formulated the NISE, or Non-Inferior Set Estimation algorithm to identify solutions in multi-objective problems which are non-inferior. This method would allow one to approximate the set of non-inferior solutions through a novel procedure. The basic idea is to approximate the weights needed with respect to a bi-objective problem such that the point defining the optimal objective in the

[13] See for example: Crowder, Johnson, and Padberg (1983) and Padberg and Rinaldi (1991) – I haven't been able to find a clear citation of the branch-and-cut algorithm but Gomory and Balas did extensive work on cutting plane methods for solving integer problems; I suspect that the branch and cut algorithm came together sometime in the early 1980's.

feasible region is determined. If we recall, a bi-objective formulation takes the form of (2.154).

$$Maximize \quad w_1 Z_1 + w_2 Z_2 \tag{2.154}$$

By approximating the lines which bound the points representing optimal solutions to each objective, the region bounding the non-dominated inferior set can be ascertained. In order to best explain the algorithm we must first define the notation that will be used in the algorithm. The set of points, P, is a set that is composed of the optimal solution points determined by the algorithm as it works through the process of finding the region defining the non-inferior set. In other words, this set contains solutions which are indexed according to the order in which they are determined by the algorithm. This means that these points may not be adjacent to neighboring points in chronological identification. In order to define solutions in order of decreasing value with respect to one of the given objectives we can reorder the set of points. This set is ordered such that each point in *S* represents the best solution with respect to the associated objective given an associated weight; this can be readily seen in Figure 2.8 where the ordering of *S* is in relation to objective Z_2. We also start with a predefined error tolerance for how accurate we wish our final result to be. To do this we use $\Psi_{i,i+1}$ which represents the error tolerance for determining the non-inferior set between solution points *i* and *i*+1. We define the maximum error tolerance as *T*. A visual representation of this is given in Figure 2.9. In order to approximate the set given in Figure 2.8 we must determine the weights that should be used with respect to each objective. We can do this through the use of weights corresponding to the line segment formed by connecting known solution

points. For example, suppose we have a known solution for the *i*-th and a neighbor represented as solution $i + 1$. The slope for the line connecting these points is given in (2.155). The slope, *m*, is:

$$m = \frac{Z_2(S_i) - Z_2(S_{i+1})}{Z_1(S_i) - Z_1(S_{i+1})} \quad \textbf{(2.155)}$$

should then be a function of the associated objective weights. This function is given as (2.156):

$$-\frac{w_1}{w_2} = \frac{Z_2(S_i) - Z_2(S_{i+1})}{Z_1(S_i) - Z_1(S_{i+1})} \quad \textbf{(2.156)}$$

where w_1 and w_2 represent weights on Z_1 and Z_2 respectively. Therefore, we can represent the weighted objective function for a segment between S_i and S_{i+1} as defined in (2.157) as follows:

$$Maximize \quad B_{i,i+1} = [Z_2(S_i) - Z_2(S_{i+1})]Z_1 + [Z_1(S_{i+1}) - Z_1(S_i)]Z_2 \quad \textbf{(2.157)}$$

Figure 2.8 - Solution Points according to set *P* and *S* as given in Cohon et. al. (1979)

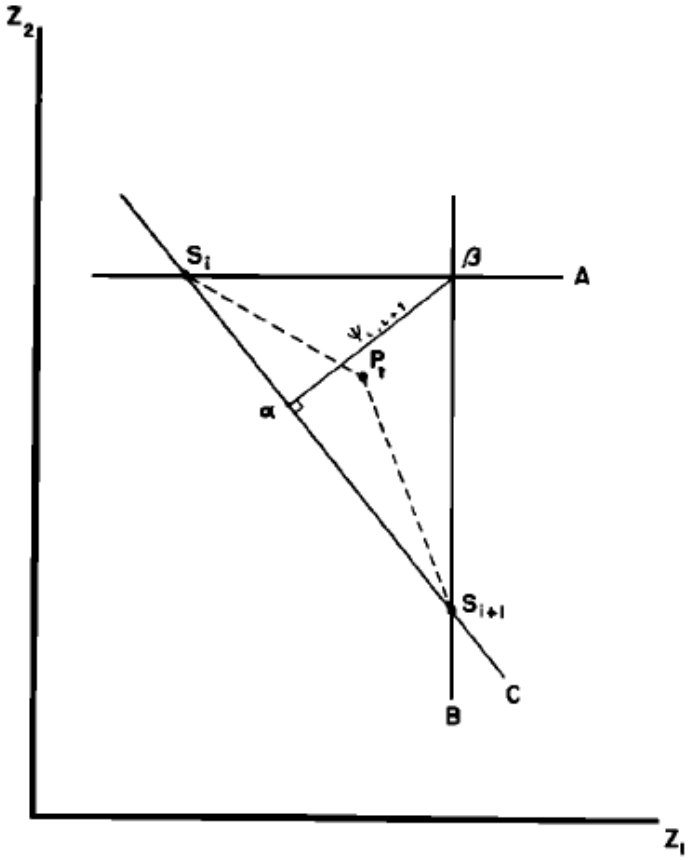


Figure 2.9 - Generating the approximate set with error tolerances as given in Cohon et. al. (1979)

through the manipulation of (2.156) and (2.154). Therefore, the generalized version of the NISE algorithm to find the approximated set is as follows:

Step 1: Set the desired value for the error threshold, *T*, the maximum allowable error. Determine P_1 by solving for Z_2 and P_2 by solving for Z_1. Set $S_1 = P_1$ and $S_2 = P_2$. Compute Ψ_{12} and let $n = 1$ where *n* equals the number of distinct points currently in the string set.

Step 2: If $\Psi_{i,i+1} \le T$ then stop as a satisfactory approximation of the non-inferior set has been determined. Otherwise, go to *Step 3*

Step 3: Find *i* such that $\Psi_{i,i+1} > T$. Solve the problem given in (156) and designate the new solution as P_{n+1}. Go to *Step 4*.

Step 4: In this step we will update the sets. Note that *i* is the highest value of *t* such that $Z_2(S_t) \ge Z_2(P_{n+1})$. Therefore, we can update the string sets in the following way:

$S'_t = S_t \quad \text{for } t = 1, 2, \ldots, i$

$S'_{i+1} = P_{n+1}$

$S'_{t+1} = S_t \quad \text{for } t = i+1, \ldots, i$

where S_t' is the new ordering for the string set. Once this is completed we now redefine the error threshold, Ψ in the following way:

$\Psi'_{t,t+1} = \Psi_{t,t+1}$ for $t = 1, 2, \ldots, i-1$ if $i > 1$

Compute $\Psi'_{i,i+1}$ which represents the maximum possible error between S_i and S_{i+1} and then compute $\Psi'_{i+1,i+2}$ which represents the maximum possible error between S_{i+1} and S_{i+2}: $\Psi'_{t+1,t+2} = \Psi_{t,t+1}$ for $t = i+1, \ldots, n-1$ if $i \leq n-2$ where $\Psi'_{t,t+1}$ are the new values for the Ψ variables. Increment *n* by one and then go to *Step 2*.

The authors note that the NISE algorithm can be modified to suit certain demands. For example, *Step 3* could be modified such that *i* can be chosen such that the maximum error $\Psi_{i,i+1}$ is the largest rather than choosing any maximum error that is above the threshold. This change would result in a more even approximation which would be useful if one is uncertain of the value chosen for the error threshold. One could also subscript the threshold itself so that different error values can be set with respect to specific segments. This would allow an analyst to gain greater insight with respect to a region of the non-inferior set which is of greater importance. To illustrate how the method works, it is useful to work through the example provided in Cohon et. al. (1979). Suppose we have a sample problem that is composed of the following objective and constraints:

$$\textit{Maximize} \quad Z = w_1 Z_1(x) + w_2 Z_2(x) \qquad \textbf{(2.158)}$$

Subject To

$$-25x_1 + 15x_2 \leq 300 \qquad \textbf{(2.159)}$$

$$-x_1 + 7x_2 \leq 300 \qquad \textbf{(2.160)}$$

$$2x_1 + 11x_2 \leq 700 \qquad \textbf{(2.161)}$$

$$x_1 + 2x_2 \leq 182 \qquad \textbf{(2.162)}$$

$$11x_1 + x_2 \leq 1120 \qquad \textbf{(2.163)}$$

$$x_1 - x_2 \leq 80 \qquad \textbf{(2.164)}$$

$$x_1 \geq 0 \qquad \textbf{(2.165)}$$

$$x_2 \geq 0 \qquad \textbf{(2.166)}$$

where $Z_1(x) = 3x_1 - x_2$ and $Z_2(x) = -x_1 + 5x_2$. The first step in the process is to identify the optimal solution with respect to each objective. This will define a set of bounds at which there can be no better solution with respect to each objective – i.e. weights are set at zero and one and the problem is solved and then weights of one and zero respectively, and then solved for Z_1 and Z_2. Thus, the first points we will add to *P* are P_1 and P_2 which define the best possible solutions with respect to each objective Z_2 and Z_1. We call these points P_1 and P_2 as these points correspond to the first two points that we have added to the set of points used to define the bounds of the solution. The third point to be found would be P_3 and so on until the algorithm terminated. If two lines are drawn with each line corresponding to the maximum value of each objective, eventually they will intersect. We can call the point of intersection between these lines, point *C*. Thus, the region formed by lines P_1C, P_2C, and P_1P_2 represent the region in which the non-inferior set must exist. In this case we will find the entire non-inferior set, so we set the error tolerance threshold equal to zero. This means that in order for the NISE algorithm to terminate all $\Psi_{i,i+1}$ must be equal to zero. Since the line P_1P_2 is not known to be inferior, Ψ_{12} is greater than zero and thus we move to *Step 3* in the algorithm. In this case, the only possible candidate is to maximize B_{12} as at this stage we only have two solution points (e.g. $n = 2$). Having made this selection we now set the point of our new solution as P_{n+1} which means we have our new solution point P_3. We now move to *Step 4* where we update the string set and redefine the parameters needed for Ψ and S. In this case, $S_1' = P_1,\ S_2' = P_3,\ S_3' = P_2,\ \Psi_{12}' > 0$, and $\Psi_{23}' > 0$. Upon completion of *Step 4* we return to *Step 2* and begin the second iteration of the algorithm. Thus, up to this point the non-

inferior set looks like that given in Figure 2.10 where the shaded region represents the non-inferior set after one iteration of the algorithm. Since we have values of $\Psi_{i,i+1}$ that are greater than zero we would need to continue the algorithm. For further examples and a complete enumerative description, see Cohon et. al. (1979).

The limitation of the NISE algorithm is that it only applies to problems which are linear in nature. This means that the optimal set of discrete points for integer or mixed integer linear formulations can be approximated but only based upon finding supported non-dominated points. Steuer and Choo (1983) utilized a Tchebycheff (also given as Chebyshev) approximation to determine the non-inferior set. This process takes advantage of the nature of Tchebycheff Polynomials by approximating the best polynomial which fits the objective for the problem. Solanki (1991) expanded the idea of by using a weighted Tchebycheff procedure for explicit multi-objective location problems. The main contribution of Solanki (1991) was to generate the non-inferior set with respect to discrete solutions. Medrano and Church (2014, 2015) have further expanded this work into finding unsupported non-dominated solutions with respect to shortest paths. These solutions are much harder to determine, although they are optimal solutions, they are not necessarily readily determined as convex supported solutions. An example of this is seen in Figure 2.11.

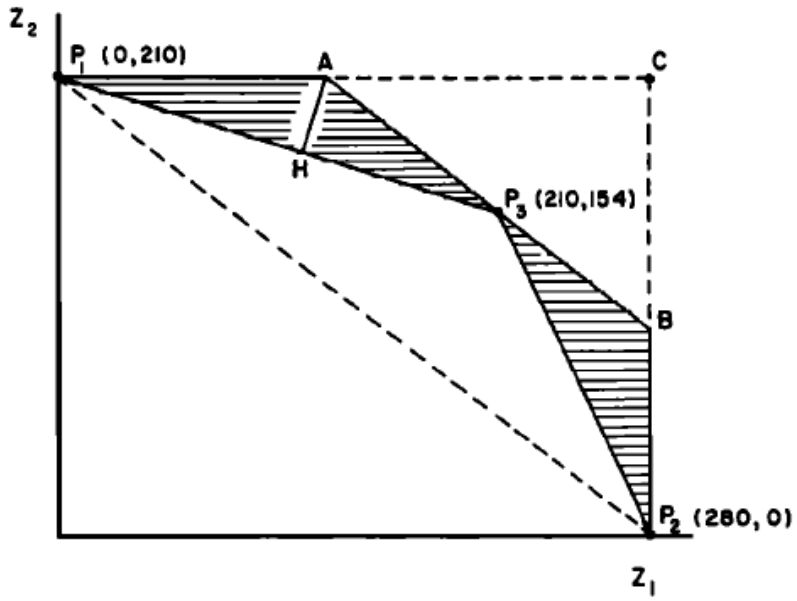


Figure 2.10 - Non-inferior set after one iteration of the NISE algorithm on the sample problem as given in Cohon et. al. (1979).

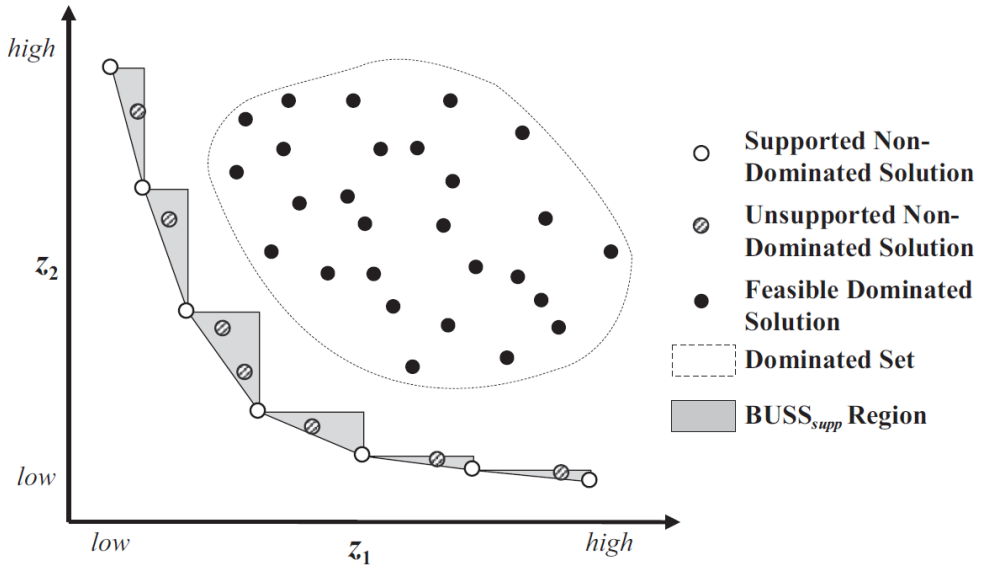


Figure 2.11 - Example of un-supported non dominated solutions as given in Medrano and Church (2014)

2.5 Heuristics

Solving large problems to optimality may be computationally unrealistic. Although solution algorithms such as branch and bound will determine an optimal solution given enough time and computational resources for many problems, there still exits a limit as to the size at which most NP-hard problems can be solved to optimality.

This necessitates the use of "good-enough" heuristics. The field of heuristic algorithm development is very large and includes techniques such as: greedy; λ-opt (swapping); insertion; GRASP (Feo and Resende, 1989); variable neighborhood search, genetic, swarm smarts (e.g. ant colony optimization); simulated annealing; heuristic concentration, Tabu search; as well as others. I will review here only those techniques that have been used in covering path problems, notably 'Tabu' search and LaGrangian relaxation. However, from the outset one should realize that all heuristic approaches are possible for application in covering path problems. The first type of heuristic applied with respect to covering paths is LaGrangian Relaxation. This will be reviewed and followed by a discussion of Tabu Search.

The first application of LaGrangian relaxation we will look it is by Current, Pirkul, and Rolland (1994) as it specifically addresses LaGrangian Relaxation as applied to the Shortest Covering Path Problem. LaGrangian relaxation is a method of relaxing selected constraints and adding "LaGrangian" terms to the objective function approximating the problem. This means that the solution obtained for the relaxed problem is an approximate solution to the original problem. Stated precisely, LaGrangian relaxation allows one to remove one or more constraints within a model by incorporating it into the objective function through the use of LaGrange multipliers, which impose costs to violations of the incorporated inequality constraints on the objective. Incorporating a set of inequality constraints into the objective and determining the proper multipliers often will allow a problem to be solved much more efficiently and easily than the original un-relaxed formulation. In practice this means that larger problems can be solved close

to optimally. Current et. al. (1994) applied LaGrangian relaxation to the Shortest Covering Path problem. If we recall that the SCP is defined as:

$$Minimize\, Z = \sum_{i \in N_j} \sum_{j} d_{ij} x_{ij} \quad \textbf{(2.22)}$$

Subject To

$$\sum_{i \in N_i} x_{1i} = 1 \quad \textbf{(2.23)}$$

$$\sum_{i \in N_n} x_{in} = 1 \quad \textbf{(2.24)}$$

$$\sum_{i \in N_j} x_{ij} - \sum_{i \in N_j} x_{ji} = 0\,, \forall j \text{ where } j \neq 1 \text{ and } j \neq n \quad \textbf{(2.25)}$$

$$\sum_{i \in N_j} \sum_{j \in s_k} x_{ij} \geq 1, \forall k \text{ where } k \neq 1 \text{ and } k \neq n \quad \textbf{(2.26)}$$

$$x_{ij} = \{0,1\}\,, \forall (i,j) \quad \textbf{(2.27)}$$

we can see that constraints of type (2.26) can be adapted into the objective function transforming the problem into a version of a shortest path problem. Thus, if we remove constraints of type (2.26) and incorporate these into our objective function using a LaGrange multiplier we will obtain the form given in (2.167):

$$Minimize\, Z = \sum_{i \in N_j} \sum_{j} d_{ij} x_{ij} - \sum_{k \in N} \lambda_k \left(\sum_{i \in N} \sum_{j \in S_k} x_{ij} - 1 \right) \quad \textbf{(2.167)}$$

where λ_k represents a vector of LaGrangian multipliers. This function can be rewritten into the form given in (2.168):

$$Minimize\, Z = \sum_{i \in N} \sum_{j \in N} \left(d_{ij} - \sum_{k \in S_i} \lambda_k \right) x_{ij} + \sum_{k \in N} \lambda_k \quad \textbf{(2.168)}$$

which is, in essence, a shortest path problem objective. For values of λ_k that are less than or equal to zero the problem can be easily solved using Dijkstra's Algorithm. However, since the LaGrangian multipliers can be greater than zero, it is possible that negative cost cycles could occur which would cause Dijkstra's Algorithm to fail as it becomes caught

in a negative cost cycle. Although other algorithms could be used – e.g. Floyd-Warshall or Bellman-Ford – Current et. al. noted that it was more efficient to discard multiplier sets which result in negative cycles. They also noted that they could have alternatively added sub-tour breaking constraints to the formulation as needed though they again noted that it was more efficient to opt to avoid sets which resulted in negative cost cycles, particularly since one searches for "good" LaGrangian multipliers.

Thus, the difficult part of any LaGrangian relaxation is finding the proper set of multipliers. This is done through the approximate LaGrangian multiplier set λ^* that is defined by (2.169):

$$\lambda^* = \underset{\lambda}{Maximize}\{Z(\lambda)\} \quad \textbf{(2.169)}$$

This function can be interpreted as follows: find the best value of lambda values such that the best objective using LaGrangian multipliers is determined in the relaxed problem. The next step is to find these multipliers. Current et. al. note that there are several methods for determining 'good' multipliers (Bazaara and Good, 1979; Fisher, 1981). The method chosen by Current et. al. is a subgradient search method. This subgradient optimization is defined by using the following function given in (2.170):

$$\lambda^{k+1} = \lambda^k + t_k(AX^k - b) \quad \textbf{(2.170)}$$

where X^k is an optimal solution to the relaxed problem and t_k is a positive scalar step size at iteration *k*. To determine the step size for the subgradient function in (2.170), the following stepsize function (2.171) is commonly used.

$$t_k = \frac{d_k(\bar{Z} - Z(\lambda^k)}{\left\|AX^k - b\right\|^2} \quad \textbf{(2.171)}$$

In this case, $\overline{Z}$ is a feasible solution value, and d_k is a scalar value such that $0 \le d_k \le 2$. To initialize the function, λ^0 values are set to zero and d_k is initialized at 2. Current et. al. halved d_k after 15 consecutive iterations when the bound did not improve. When multipliers were found that resulted in negative cost cycles, Current et. al. discarded these multipliers, halved d_k and resumed the optimization using the multiplier set that generated the best bound found up to that time. In order to generate feasible solutions to the relaxed SCP, Current et. al. formed a heuristic which would take a solution to a multiplier set and determine if the solution covered all nodes. If so, a feasible solution had been determined. If not, the following heuristic was employed.

Step 1: Let "path" be the path identified by the solution to the relaxed SCP. Initialize *D* to be the set of all nodes not on the path, and let *U* be the set of all nodes not covered by the "path." If $U = \{\phi\}$ stop: "path" is a feasible solution.

Step 2: Find the node *q*, where $q \in D$, which covers the maximum number of uncovered nodes. If $D = \{\phi\}$ then stop; no feasible solution is determined.

Step 3: Find two nodes *k* and *l* which are vertices of an arc on the "path" such that $d_{qk} + d_{ql} - d_{kl} \le d_{iq} + d_{qi} - d_{ij} \quad \forall i, j$ where *i* and *j* are vertices of an arc on the "path." If no arc exists connecting node *q* with nodes on the "path" then let $D = D - \{q\}$ and go to step 2. Otherwise, insert node *q* on the "path" between *k* and *l* and update *U* and *D*.

Step 4: if $U = \{\phi\}$ then stop; a feasible solution has been identified, otherwise go to step 2.

Essentially this heuristic takes a solution obtained from LaGrangian multipliers in the relaxed SCP and checks to see if all nodes are covered. If there are any uncovered nodes, then nodes are inserted into to the path in a greedy manner. This process is then repeated until there are no uncovered nodes or a solution has been determined to be infeasible. This is done in order to find a feasible, but not necessarily optimal, solution to the problem so that one can begin to optimize along the subgradient. Thus, the application of LaGrangian relaxation to the SCPP allows the problem to be solved quite quickly and in

this case without the addition of sub-tour elimination constraints. Although no commentary was given with respect to loops occurring in solutions (with the exception of negative cost cycles) in their LaGrangian relaxation based approach, the process defined by Current et. al. does not generate paths with loops. When a loop does exist in an optimal shortest covering path, however, their process will not completely converge. Furthermore, if the truly optimal solution includes negative multipliers, it is possible that the true optimal primal solution could be overlooked. The drawback to using LaGrangian relaxation lies in the fact that a formulation must be set up in such a way that the use of a LaGrangian relaxation makes sense. In other words if you are attempting to reduce the constraint size of a problem, the constraint to be relaxed should be one of sufficient size and of an appropriate form.

However, Fernandez and Marin (2003) formulated a LaGrangian relaxation based heuristic for what they called the Path Location with Multi Source Demand Problem. The stated goal of that problem is that it should be applied to situations and applications in "which the demands of one client must be satisfied through a set of different service points (Fernandez and Marin, 2003)." In essence they are trying to model how to best locate stops such that the site locating cost and the overall route cost (i.e. distance or time) is minimized while user utility is maximized. The main value in this work, however, is the fact that their problem can be formulated in such a way that a MCSP can be subsumed and a LaGrangian relaxation applied. It is important to note several major assumptions in their work. The first is that the network is directed, the second is that there exists costs and demands associated with each node, and finally – and most importantly – the route is acyclic; in other words it must not contain *any* loops or cycles.

The reason for this is exactly the same as in the relaxed SCP; a sub-tour can involve a negative cost cycle resulting from negative multipliers. Thus, the possibility of a loop occurring because of a negative cost cycle may imply that an optimal solution might indeed include a loop. By assuming that an optimal solution must be acyclic and that such conditions be ignored means true optimal solutions may not be generated and identified. Nevertheless, when applied properly, LaGrangian relaxation based approaches are very powerful tools that can often find good bounds on optimality as well as feasible, high performing solutions[14].

The last type of heuristic we will examine is Tabu Search. Tabu Search was developed by Glover (1989); the basic idea of Tabu Search is to overcome local optima through a search procedure which forces solutions to be excluded from a candidate set for a period of time, usually a set number of iterations. The idea is based upon the idea of a hill climbing heuristic, or a procedure which looks for the best solution and which 'climbs' the optimal curve. The simplex method (Dantzig, 1951) is an example of a type of climbing algorithm as it pivots from extreme point to extreme point along the feasible region in an improving direction, working its way to the optimal solution. However, in the case of a non-convex feasible region, it is possible that a climbing heuristic may be stuck in what is called a local optima. A local optima is optimal with respect to a localized area within the feasible region but in the entirety of the solution space is not the optimal solution. In an illustrative example, one can think of a local optima as a foothill on the way up to the tallest peak of a mountain range. In a minimization context, one can think of the 'climbing' process as descending a hill and looking for the lowest possible

[14] It is important to note that it is possible to embed a LaGrangian approach in a Branch and Bound algorithm and identify and verify optimal solutions for certain classes of integer programming problems.

valley. Thus, any climbing heuristic has the potential to enter a local optima and define this as an optimal solution even though a better solution may occur in a different part of the region of feasibility.

Tabu Search is a type of meta-heuristic in which a search for an optimal solution is initiated using another heuristic search process, but tries to ensure true optimality by forcing certain solution elements to be 'Tabu' in order to avoid getting stuck at a local optimum. In order to give the basic algorithm it is prudent to define the key notation. The general heuristic and notation is as follows in a minimization context:

X = the set of solutions for the problem
x = discrete solution in X
S = the set of moves one makes towards an optimal solution
s = the current move from one solution to another
T = the set of 'Tabu' solutions which is a subset of S
t = the number of iterations that a solution is to remain in T
$OPTIMUM$ = an evaluator function for the solution

Step 1: Select an initial solution such that $x \in X$ and let $x^* = x$. Set the iteration counter $k = 0$ and begin with the 'Tabu' set, T, to be null (ϕ).

Step 2: If $S(x) - T$ is null then go to *Step 4*; otherwise, set $k = k + 1$ and select $s_k \in S(X) - T$ such that $s_k(x) = OPTIMUM(s(x) : s \in S(x) - T)$.

Step 3: Let $x = s_k(x)$. If $c(x) \le c(x^*)$, where x^* is the best solution currently found, then set $x^* = x$

Step 4: If a chosen number of iterations has elapsed either in total or since x^* was last improved, or if $S(X) - T = \phi$ upon reaching this step from *Step 2*, stop. Otherwise update T and return to *Step 2*.

The way Tabu Search works can be described as follows. In *Step 1* we need to identify an initial starting solution that must be feasible to the stated problem. Although not stated explicitly in *Step 1* this solution can be derived through the use of another heuristic, such as greedy, etc. One could even fill in the elements randomly; the point is that we simply need to determine a solution that is feasible in order to begin the Tabu Search process. We then set this solution to be the best solution currently found, x^*.

Since this is the first initial step in starting Tabu Search we set the iteration counter to be equal to zero, and the 'Tabu' set to be null as we have not presently added any elements in this set. *Step 2* first checks to see if the set of solution elements minus 'Tabu' elements is null. If it is null the heuristic goes to *Step 4*. If one has just initialized the process this set is not empty as set $S(x)$ will be comprised of the initial solution. Since the 'Tabu' set is null then by evaluation of $S(x)-T$ we determine that the set is not empty, and therefore we increment k by one and find another solution to the problem through some function. This function is defined above as *OPTIMUM*; the *OPTIMUM* function can be defined in several ways. One possible method is through the use of some kind of heuristic – depending on the problem this could be a swap heuristic or something similar – or by solving an 'auxiliary optimization problem' (Glover, 1990a). Typically the reason for solving an auxiliary problem is to enable one to determine a feasible starting solution; for example this can be used in Dantzig's Simplex Algorithm to specify a feasible starting point. In this case an auxiliary problem is used in Tabu Search with respect to IP/MIP problems and is detailed extensively in Glover (1990a). To keep things understandable here, one can think of the goal of the *OPTIMUM* function as a way of finding a better solution, given that there are some 'Tabu' restrictions imposed.

In *Step 3* we let the solution we obtained in *Step 2* be represented by x. If the objective value of this solution, e.g. $c(x)$, is less than the objective value for the current best solution, e.g. $c(x^*)$, then we set the current best solution to be equal to x and then move to *Step 4*. In *Step 4* we keep track of what has happened with respect to the search process. If we have reached a certain number of iterations, e.g. $k = p$ where p is the maximum number of iterations, then the algorithm stops. Similarly, if the best solution

found, e.g. $x*$, has not changed for a certain number of iterations then the algorithm ends. The final stopping condition, where the set of solutions minus the 'Tabu' set is null, implies that there are no moves which can improve the solution and thus the heuristic ends. If this is not the case, i.e. none of the stopping conditions are met, then the 'Tabu' set is updated and the algorithm repeats at *Step 2*. The remaining question then becomes: what criteria is used to update the 'Tabu' set? A generalized method of adding solutions to the 'Tabu' set is given in (2.172). In essence, this function represents the set

$$T = \{s^{-1} : s = s_h \quad \text{for } h > k - t\} \qquad \textbf{(2.172)}$$

of 'Tabu' solutions which would 'undo' the current solution. In other words, the 'Tabu' set is used to prevent solutions from returning to a previous solution state. Thus, we don't want to make moves which will satisfy $s^{-1}(s(x)) = x$. In order to track changes over time we introduce the index *t* which represents the number of iterations a move has been in the 'Tabu' set. Therefore, (2.172) can be evaluated as follows: The 'Tabu' set will consist of all moves which enable a previous solution such that this move occurred within *t* iterations of the 'Tabu' Search heuristic. For example, if *t* was set to be equal to 10, then all previous solutions determined in the previous ten iterations would be included in the 'Tabu' set. By setting a 'Tabu' set, we can ensure that a variety of differing solutions are obtained, while also allowing a previous solution element to exit the 'Tabu' set after a certain number of iterations to avoid the case of entering into a local optima. It should be noted that the method of adding elements into the 'Tabu' set given in (2.172) can vary; Glover (1989, 1990a) gives several alternate ways of adding and removing elements from the 'Tabu' set. I have only given the generalized form as it

describes the function of the heuristic quite well. There are also several very good instructional papers and books such as Glover (1990b), Glover and Taillard (1993), Gendreau (2003), Glover and Laguna (2013) as well as many applications based papers – i.e. Tabu Search for *p*-median (Rolland, et. al., 1997) – all of which are too numerous to note here.

There are a number of papers that have direct links to covering-path type problems that should be noted here. Gendreau et. al. (1995) utilize Tabu Search for the location of rapid transit lines, while Dufourd et. al. (1996) use Tabu Search to locate a transit line. Fan and Machemehl (2004) use a Tabu Search procedure to solve the network design problem and Fan and Machemehl (2008) utilize Tabu Search to optimize public transportation networks with variable transit demand. Although there are no papers that deal directly with a Maximum Covering Shortest Path problem, there is no reason that 'Tabu' Search could not be an effective strategy for finding a good solution. Since 'Tabu' Search is a meta-heuristic, that is it can employ other heuristics to obtain starting results and potentially different solutions, it is possible to employ other more basic heuristic methods to aid in the search for optimal solutions, while Tabu search controls the process. The biggest issue here is that no basic heuristic has been developed that involves solving path like problems that expressively allow loops to be embedded in their solutions.

2.6 Conclusions

This chapter has examined the underlying models which form the basis of covering-path problems. Namely these include the Location Set Covering Problem (LSCP), the Maximal Covering Location Problem (MCLP), and the Shortest Path

Problem (SPP). We have examined how these base models were developed into the Shortest Covering Path and Maximum Covering Shortest Path problems (SCP, MCSP) as well as several related problems such as the Median and Maximal Covering Tour Problems (MTP/MCTP), the Covering Salesman Problem (CSP), the Hierarchical Network Design Problem (HNDP), the Transit Arc-Node Service Maximization Problem (TRANSMax), the Minimum Covering Shortest Path Problem, and several multi-path covering problems. We have also examined algorithmic procedures used to find exact solutions to covering-path problems as well as several heuristic based approaches which are used to solve covering-path type problems such as Branch-and-Bound, Non-Inferior Set Estimation (NISE), LaGrangian Relaxation, and Tabu Search. We have also examined several issues with respect to covering-path type problems such as the methods used to eliminate sub-tours as well as problem specific issues such as assumptions related to directionality and service.

This examination has shown that there are deficiencies with respect to how models describe the real world, how these problems are complex, and how these problems can potentially be solved using algorithmic/heuristic procedures. Chapter 3 of this dissertation will explore how covering-path models can be developed to reflect real world routing situations. In particular it will address how the MCSP and MPSP models can be reformulated to be 'loop agnostic.' Chapter 4 presents a new heuristic which can calculate good, feasible solutions to the New, Revised MCSP problem formulated in Chapter 3. Chapter 5 will present a new model formulation based on the alternatively formulated MCSP, TRANSMax. Particular attention is given to how the model can be reformulated to be loop agnostic as well. Chapter 6 will then build upon these new loop

agnostic models and present the new Bi-Directional Maximal Covering Shortest Path problem which accounts for travel in opposite directions. The models formulated to address the BD-MCSP are not only loop agnostic, they also allow unique forms of loops such as the bi-directional and uni-directional loop structures.

Chapter 3

3.1 Introduction

The maximal covering shortest path problem is a bi-objective problem that involves the location of a path connecting an origin with a destination where path distance is minimized and path coverage is maximized. The original model formulation for this problem was developed by (Current et al. 1985) and was cast as an integer-linear programming problem. Because this is a discrete optimization problem, solutions in objective space are represented as discrete points (Cohon, 1978). Pareto-optimal solutions are those discrete solutions where it is impossible to increase path coverage without increasing path distance or where it is impossible to reduce path distance without decreasing path coverage as well.

Current et al. (1985) solved their model using a weighting approach described in Cohon (1978), which is capable of identifying all supported, Pareto optimal solutions in objective space. The shortest covering path problem is a special case of the maximal covering shortest path problem, and an optimal solution to the SCP is the most efficient "cover all nodes" path and is a supported Pareto-optimal solution to the MCSP. All other Pareto-Optimal solutions to the MCSP cover less than all nodes and have a path distance less than any SCP and are therefore infeasible to the SCP. Since Niblett (2013) demonstrated that the original model to the SCP problem may not in all cases identify an optimal solution, it is then possible that solutions to the MCSP of Current et al. may not always be optimal in the sense of Pareto-optimality. The reason for this is that Pareto-optimal solutions to the MCSP may involve embedded loops. From a technical point of

view, such solutions may not be considered paths; however, Minieka (1978) defined a shortest path as the path that has the smallest possible length. Consequently, it is the shortest path that accomplishes the desired task, whether it contains a loop or not. In addition, some approaches to the k^{th}-shortest path problem expressively allow for the construction and use of loops (Dreyfus, 1969; Shier, 1979; Martins, 1984; Eppstein, 1998). This means that path definitions and solution algorithms for different types of path problems have included the possibility that loops may be used as part of a solution.

The original shortest covering path model of Current et al. (1984) was predicated upon an implied assumption that optimal solutions would *never* contain a loop or return to a previously visited node. This assumption seems straight forward, except for the simplest type of networks. For example, when turn restrictions are encountered at intersections on a road network, some shortest paths leave a specific intersection in the straight direction and then make a series of three right turns at subsequent intersections in order to approach and go through that intersection for the second time to overcome the case where a left hand turn was prohibited on the first approach and prevented a direct left-turn move for that desired direction. Thus, in turn-restricted networks, loops may be somewhat common in shortest paths.

This chapter introduces revised models that have been developed for the Maximum Covering Shortest Path (MCSP) problem and the Maximum Population Shortest Path (MPSP) problem with the intention of allowing loops to occur when it is optimal to use them. Such models should be considered to be loop agnostic – loops are neither required nor prevented. This chapter also presents computational experience in solving these problems as well as several example solutions that demonstrate how the

new, revised models can find solutions that are as good as, or exceed, those solutions produced by the original models of Current et al. (1985). This chapter expands the work of Niblett (2013) and Niblett and Church (2016), where they analyzed issues with respect to the Shortest Covering Path problem and offered a new, revised formulation of the SCP. Here, it is shown that the original MPSP model formulation may result in the case that no feasible solution exists, when in fact a feasible solution does exit.

This chapter is structured as follows: the next section will provide an overview of the new, revised formulation of the maximum covering shortest path problem. Key nuances in the formulation will be discussed, particularly how they compare to those in the classic MCSP formulation given in chapter 2. The third section will discuss the computer environment in which the NR-MCSP formulation was applied as well as discuss the workflow of how the problems were solved. The fourth section will present results from the model and compare them to counterpart solutions to the original MCSP model. The fifth section addresses the related maximum population shortest path (MPSP) model developed by Current et. al. (1985), particularly in how the original model may be incapable of finding an optimal solution. Results from the original MPSP as well as those derived utilizing the EAST constraint process described in Chapter 2 in conjunction with a new revised MPSP model will then be compared. The sixth and final section will offer concluding remarks.

3.2 A New, Revised Model for the Maximum Covering Shortest Path Problem

The formulation for the classic MCSP proposed by Current, et. al. (1985) is given in Chapter 2; the model presented here is an expansion of this formulation and is setup in

such a way that Eliminate or Attach Sub-Tour (EAST) constraints can be utilized. As in Chapter 2, we assume that there is a network of nodes and arcs. Arcs may be directed or undirected. The first problem we will address is a new, revised formulation for the MCSP. The notation and formulation for the NR-MCSP problem is as follows:

$i, j, k =$ indices used to reference nodes of the network

$a_k =$ the amount of demand at node k where $a_k \geq 0$

$\alpha =$ the importance weight associated with the coverage objective

$\beta =$ the importance weight associated with the distance objective

$d_{ij} =$ the distance or travel time from node i to node j

$x_{ij} = \begin{cases} 1, \text{if an arc from } i \text{ to } j \text{ is traversed from } i \text{ to } j \text{ in the MCSP} \\ 0, \text{otherwise} \end{cases}$

$y_k =$ one if node k is covered and zero if it is not

$p =$ the starting node for the shortest covering path

$q =$ the terminus node for the shortest covering path

$N_j = \{ i | \text{arc}(i, j) \text{ exists } \}$ the set of nodes i which are connected to j

$S^* =$ the maximum allowable service distance/time

$S_k = \{ j | d_{jk} \leq S^* \}$ the set of nodes j which are within the maximum service distance to node k

$F_i =$ the set of nodes, k, that are connected to i by an arc and can be directly traversed from i to k. $T_j =$ the set of nodes, k, that are connected to j by an arc and can be directly traversed from j to k.

$Z_C = \alpha \sum_k a_k y_k$ the objective term corresponding to coverage

$Z_D = \beta \sum_{i \in N} \sum_{j \in F_i} d_{ij} x_{ij}$ the objective term corresponding to path distance

$$\text{Maximize} \quad Z_C - Z_D \tag{3.1}$$

Subject to:

$$\sum_{j \in F_p} x_{pj} - \sum_{i \in T_p} x_{ip} = 1 \text{ for origin node } p \tag{3.2}$$

$$\sum_{i \in T_q} x_{iq} - \sum_{j \in F_q} x_{qj} = 1 \text{ for destination node } q \tag{3.3}$$

$$\sum_{i \in T_k} x_{ik} - \sum_{j \in F_k} x_{kj} = 0, \forall k \in N \text{ where } k \neq p \text{ and } k \neq q \tag{3.4}$$

$$\sum_{i \in N_j} \sum_{j \in s_k} x_{ij} - y_k \geq 0 \,, \forall\, k, k \neq p, q \quad \textbf{(3.5)}$$

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij} - \sum_{i \in V} \sum_{\substack{k \in T_i \\ k \notin V}} x_{ki} \leq |V| - 1 \,, \forall \text{ subset of nodes } V, \text{ when } |V| \geq 2 \quad \textbf{(3.6)}$$

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij} - \sum_{i \in V} \sum_{\substack{k \in F_i \\ k \notin V}} x_{ik} \leq |V| - 1 \,, \forall \text{ subset of nodes } V, \text{ when } |V| \geq 2 \quad \textbf{(3.7)}$$

$$x_{ij} = (0,1) \,, \forall (i,j) \quad \textbf{(3.8)}$$

$$y_k = (0,1) \,, \forall\, k \quad \textbf{(3.9)}$$

The objective (3.1) remains the same as in the original MCSP; the first term is a weighted objective term that involves maximizing population covered and the second term is a weighted objective term that involves minimizing overall path length. Constraint (3.2), however, is modified from the original MCSP formulation. Constraint (3.2) allows for a loop to form at the origin node so long as there is one more arc used to leave the origin node than is used to enter it on the covering path. This is done by ensuring that the sum of all arcs used to leave the origin must be one more than the sum of the arcs used to enter the origin. This form of (3.2) allows for a 'lollipop' form of route to occur where a loop is attached at the origin node. In a similar vein, Constraint (3.3) allows a loop in the path to involve the destination node. The path may, essentially, loop through the destination node, but it must eventually end at the destination. This is ensured by the fact that the path must enter the destination node exactly one more time than the path departs from the destination node. This constraint allows a 'lollipop' form of looping path to occur at the destination (one or more times if necessary). Taken together, constraints (3.2) and (3.3) allow for the possibility of a 'barbell' like route to occur, where loops are added at the start and end of the route resulting in a barbell-esque route appearance.

Constraints (3.4) are classical balance constraints which are unchanged from the original MCSP; if an arc is used to enter node *k* then a subsequent arc must also leave node *k*. These constraints are imposed for all intermediate nodes, that is, for all nodes except the origin and destination nodes. Constraints (3.5) are also unchanged from the original MCSP. These constraints define whether a node is covered by the path; if the path enters a node *j* which is within the maximum service distance of node *k* then node *k* can be considered covered. Constraints (3.6) are the EAST constraints developed by Niblett and Church (2016). These constraints ensure that a sub-tour connecting the set of nodes *V* cannot exist in the solution unless there is an arc used to enter the sub-tour that is not part of the sub-tour itself. If the sub-tour is not entered by an external arc – i.e. an arc which is not part of the sub-tour – then the constraint devolves to the traditional Dantzig, Fulkerson, and Johnson sub-tour breaking constraint and the tour cannot exist in the solution. However, if an external arc is used to enter the loop then the loop is allowed to occur as part of the covering path. Constraints (3.7) are similar to (3.6) but in this case the tour is allowed to form only when an external arc which is not part of the sub-tour is used to depart that sub-tour to some other node $\notin V$, otherwise the sub-tour cannot form in the solution. Taken together, these two constraints then force a sub-tour to be connected to other nodes in the path which are not a part of the sub-tour or the sub-tour cannot be used in the solution.

It should be noted that the number of constraints (3.6) and (3.7) increase exponentially while increasing the number of nodes *n* comprising the network as there are two constraints for every possible subset $V \subset N$, where $|V| \geq 2$ and $|V| \leq n-1$. Because it is not practical to enumerate the complete set of constraints *a priori*, these

constraints are added to the problem whenever a disconnected sub-tour appears in a solution. This model is then re-solved until no further disconnected loops or sub-tours appear in the solution. Constraints (3.8) and (3.9) list the necessary binary constraints on the decision variables. The key point in the formulation of this new, revised model is that an attached loop is not expressively prevented from being used in a solution. That is, the model is loop agnostic – loops can be used if their presence results in an optimal solution. In the next section we show that loops/attached-tours may, in fact, be part of an optimal covering path solution.

3.3 Applying the NR-MCSP Model

The mathematical model given above was formulated and solved using the Xpress-IVE mathematical programming and modeling environment (version 7.8). This software is marketed and maintained by FICO. Models were written in Mosel, a modeling language provided in the Xpress-IVE 7.8 environment on a PC operating Windows 7 Professional. Although the Xpress solver is included within the Xpress-IVE modeling environment installed on the PC, all solver computations were accomplished using the Xpress solver within the Ubuntu 14.04 Long-Term-Support server operating system. The specifications for the Ubuntu Linux system are as follows: two Intel Xeon X5560 CPUs which in total provide 8 logical cores and 8 virtual (hyper-threaded) cores; a total of 48GB of DDR3 RAM running at 1333 MHz; and 136GB of hard disk drive space using SAS SSD drives configured in a RAID 5 storage array.

The workflow for modeling the NR-MCSP is shown in Figure 3.1. It should be noted at the start that the workflow is designed so that a series of problems (one for each desired maximal service distance) are solved automatically. The steps in modeling are as

follows: 1) Obtain spatial data and input this information into associated data files which can be read by Mosel; 2) Setup the model using the maximum service distance that one wishes to solve and compile the problem written in Mosel so that it can be read by the Xpress solver. It should be noted that when solving the MCSP and NR-MCSP models we *a priori* added associated Dantzig, Fulkerson, and Johnson or EAST constraints respectively for 'simple tours' or what we call out and back loops – that is, loops which can form between two adjacent nodes that are connected by an arc. These constraints are easy to add and are relatively small in number, equaling at most 2 times the number of arcs. Simple disconnected loops can often appear in solutions to problems without any loop restrictions, and adding these conditions can be considered a preemptive strategy. However, complex sub-tours are likely to be found in the solution and still need to be accounted for. In this case, two EAST constraints are required as compared to the Dantzig, Fulkerson, and Johnson inspired sub-tour elimination constraints utilized by Current, et. al. Two EAST constraints are required as we need to add a constraint for both entering the sub-tour as well as leaving the sub-tour.

We found that an *a priori* addition of these simple sub-tour elimination constraints had no notable increase in solution time beyond solving a model without such constraints; however, doing so can eliminate a number of iterations needed to generate a feasible, optimal solution to their respective problem (MCSP or NR-MCSP). Once the model and associated constraints are compiled, the workflow proceeds to step 3). In step 3) the Xpress solver is utilized to solve the current model instance; once the solver outputs a solution, workflow continues to step 4) where a check is made to determine if any sub-tours exist in the solution. This step is coded entirely in Mosel. This detection

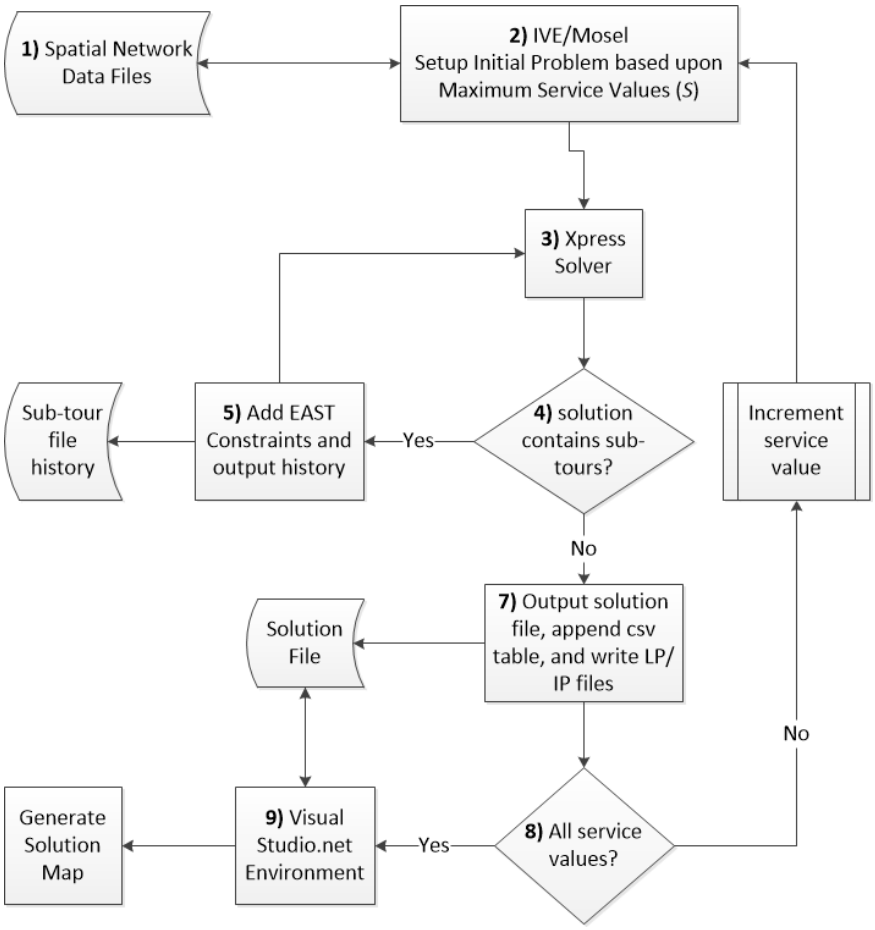


Figure 3.1 - Workflow in determining and displaying optimal solutions to the NR-MCSP problems

routine utilizes a node coloring approach and is structured in the following way. The routine first begins with the origin node, p, and sets it to be the 'from node' and checks all arcs which can be directly traversed from the 'from node' to other nodes. If an arc from the 'from node' to a node k is used in the solution then node k is colored with a temporary label and placed in the set of nodes that are labeled as temporary. Once all arcs from the origin have been checked, we color this 'from node' as being permanently labeled. The search process then checks if there are any nodes that are currently labeled as temporary. If there is a node in the set of temporary labeled nodes then one such node is removed from this set as it is selected as the new 'from node' and the process of coloring new temporary nodes that are reached from this new 'from node' using path arcs

is repeated. Overall, this process is continued until no nodes are present in the set of temporary labels. We then save the set of colored nodes as these represent nodes which are directly visited by an arc on the covering path. The solution is then checked against the set of colored nodes; any nodes which have not been colored then represent nodes which lie on one or more sub-tours. If there are no sub-tours detected then the solution file for the problem is output along with the LP/IP model formulation and a comma separated value (CSV) file is appended with the optimal solution objective value, total coverage, path distance, and maximum service value (Step 7).

If it is determined that a sub-tour or a set of sub-tours are present in the solution, we need to define each sub-tour with respect to the nodes and arcs which form the sub-tour. This is represented as step 5) in the workflow. To define the set of complex sub-tours, we select the first node that is found in the solution set which has not been permanently colored and set this as the new 'from node'. At this point a new color is selected, the 'from node' is colored as permanent, and the process continues anew to color all nodes reached by arcs used in the solution from this new start of the coloring process. This procedure is repeated until all nodes in the solution have been permanently colored. Thus, this process first determines the path that connects the origin and destination nodes along with any connected loops to this path. Then it determines all sub-tours or loops that are part of a solution. All loops/sub-tours (nodes and arcs) that are found are stored on a list. Once this process has completed, the model is then modified by either adding one Dantzig, Fulkerson, and Johnson tour-breaking constraint for each sub-tour encountered in the case of the MCSP or by adding a pair of EAST constraints for each sub-tour encountered in the case of the NR-MCSP.

Once these constraints have been added and an associated log file has been written, this new modified model is solved by returning to 3). The process loops through steps 3 and 4 until enough constraints have been added so that the resulting solution is sub-tour free. If this is the case then the solution file is written as shown in step 7). In step 8), the current value of the maximum service value *S* is checked; if it is not the last maximum service value to be run, the process returns to step 2) and initializes a new problem with the next maximal service distance to be considered. Otherwise, if solutions for all desired maximum service values have been computed the results are then processed by a program that was developed in Microsoft's Visual Studio.net SDE represented in step 9) of the workflow. This program produces a map of the solution on the associated network, displays this solution map on a visual interface, and then saves an image of the map. This program is designed to read the solution files produced by the Xpress-solver. The program also reads in the associated spatial network and then parses the solution file and generates a graphical map of the results. Once this has been generated a graphic solution image can then be saved as a common *.jpg, *.png, or *.bmp image file.

3.4 Results and Comparison of the NR-MCSP to the MCSP

The framework described above was used to solve and develop images of solutions for both the MCSP and NR-MCSP models. The new, revised maximal covering shortest path model formulation and the original maximal covering shortest path problem model formulation were applied to a network form of the well-known Swain dataset. This network and associated points are shown in Figure 3.2. The network of arcs was formed randomly, where arcs tend to connect near-by nodes in order to mimic

possible connections descriptive of conditions found in real world networks. Both the MCSP and NR-MCSP model formulations were solved with a series of different maximum service values and objective weights in order to develop a large, representative set of solutions. The complete table of these results can be found in Appendix I. Both the New, Revised MCSP and the classic MCSP were solved using maximum service values ranging from zero to thirty-five in increments of 2.5 as well as an additional service value of 50. The additional service value of 50 was used to compare cases where the solution to the shortest path would also provide complete coverage for all nodes in the Swain dataset. Importance weights were defined to range between 0 and 1 in increments of 0.01 (where $\alpha = 1 - \beta$) in order to determine a relatively complete set of supportive non-dominating solutions; a cover value of 1 was approximated using a weight of 0.99 and a distance weight of 0.01 as a distance weight of zero results in an impractical solution.

There exist many supportive, non-dominating solutions to this multi-objective problem. To demonstrate this for the Swain network, Figure 3.3 presents a tradeoff curve of solutions generated by both models when the maximal service distance is zero. Solutions that are depicted with circles are optimal solutions to the original MCSP problem and those that are solutions to the new NR-MCSP model are depicted with plus signs. These solutions are the unique solutions generated by the range of weights used for each model. Note that solutions to both problems trace out a similar curve, but that the number of uniquely different solutions identified as non-dominated is larger for the NR-MCSP than what was found for the MCSP. Also note that in the region of the "elbow" of the tradeoff curve, the number of solutions for the NR-MCSP is more numerous than for

the original MCSP. For this tradeoff, all problems were solved utilizing an origin node of 27 and a destination node of 21. Table 3.1 contains all of the unique results for the new, revised MCSP with a service distance of zero and Table 3.2 contains all of the unique results for the MCSP with a service distance of zero. For example, note in Table 3.1 that the first line details a solution generated for a distance weight of 0.01. The second line given in the table is associated with a distance weight of 0.18. This means that distance weights between .01 up to and including 0.17 resulted in the same solution as that generated for the distance weight of 0.01. Many of the solutions in this table involve the use of one or more attached loops as part of the optimal path. The gray-shaded rows indicate when this occurs. Thus, 8 out of 24 supported solutions involve the use of loops. When this occurs, such solutions eclipse the performance of the solutions of the classic MCSP, albeit such differences tend to be small in absolute value.

One should recognize that if a high relative weight is placed on path distance as compared to path coverage, solutions tend to be very efficient and probably will not double back with the exception of the case where a maximum covering distance of zero is used. This is primarily due to the fact that this means the path must directly visit a node which is part of the path and thus a loop which doubles back may in fact be an optimal route. However, as objective weights begin to emphasize total coverage over path-length solutions tend to utilize loops. This is not surprising as we expect that loops/tours can, in fact, be used as the overall gain from the use of a loop may result in a solution which provides more cover per unit of path length. Solutions with a maximal service distance of zero are also optimal solutions to the Maximum Population Shortest Path problem, which will be discussed shortly.

Table 3.1 – Unique solutions for the NR-MCSP where the maximal service distance was equal to zero. The table includes coverage weights, distance weights, total path length, and path coverage for each unique solution found.

Distance Weight	Coverage Weight	Total Path Distance	Total Path Coverage
0.01	0.99	312.11	640
0.18	0.82	297.55	637
0.21	0.79	269.82	630
0.27	0.73	236.73	618
0.30	0.70	224.96	613
0.34	0.66	219.12	610
0.39	0.61	208.04	603
0.43	0.57	202.68	599
0.44	0.56	198.81	596
0.47	0.53	185.08	584
0.50	0.50	156.15	556
0.51	0.49	144.34	544
0.56	0.44	119.83	513
0.57	0.43	112.90	504
0.58	0.42	97.08	483
0.62	0.38	90.83	473
0.67	0.33	86.32	464
0.72	0.28	75.45	437
0.76	0.24	68.94	417
0.89	0.11	62.47	366
0.90	0.10	58.47	331
0.95	0.05	54.00	260
0.97	0.03	52.06	206
1.00	0.00	51.92	171
***Note that shading denotes a solution where a loop has been attached to the covering path and represent an integral part of the optimal solution**			

Table 3.2 – Unique solutions found for the MCSP where the maximal service distance was set to zero. The table includes coverage weights, distance weights, total path length, and path coverage for each unique solution found.

Distance Weight	Coverage Weight	Total Path Distance	Total Path Coverage
0.01	0.99	312.80	640
0.27	0.73	236.73	618
0.43	0.57	202.68	599
0.47	0.53	185.08	584
0.49	0.51	185.08	584
0.53	0.47	144.34	544
0.56	0.44	119.83	513
0.57	0.43	119.83	513

Distance Weight	Coverage Weight	Total Path Distance	Total Path Coverage
0.58	0.42	104.01	492
0.64	0.36	97.76	482
0.67	0.33	93.25	473
0.75	0.25	78.71	443
0.84	0.16	61.07	352
0.85	0.15	61.07	352
0.89	0.11	58.47	331
0.99	0.01	52.06	206
1	0	51.92	109

An example of an optimal solution which embeds a loop within the covering path is given in Figure 3.4. This path is listed in Table 3.1 where routes involve a coverage distance of zero. Note that the path includes an embedded tour from node 1 to node 5 to node 11 to node 13, and then back to node 1. Such tours are also utilized in cases where maximum service values are non-zero – i.e. where a node need not be directly visited by the path to be covered. An example of a loop/tour being used for a positive maximum service distance is given in Figure 3.5; in this case this optimal covering path was formed for a maximum service distance of 10, using a distance importance weight of 0.14 (with a resulting coverage weight of 0.86). Note that the covering path solution utilizes a loop/tour that travels from node 53 to node 50 and back again to 53. The tables of unique non-dominating optimal solutions to the NR-MCSP and MCSP that involve a maximum service distance of 10 are given in tables 3.3 and 3.4. Based upon these results it is clear that loops/tours can and do appear in optimal covering paths and that they help to efficiently increase coverage per unit of path distance. Such solutions occur for maximal covering distances of zero as well as maximal covering distances that are greater than zero.

As mentioned above, all solutions for the test problems are given in Appendix I. Based upon the entire set of solution data presented in Appendix I that were generated

Table 3.3 – Unique solutions to the NR-MCSP using a maximum service distance of 10. The table includes coverage weights, distance weights, total path length, and path coverage for each unique solution found.

Coverage Weight	Distance Weight	Total Path Coverage	Total Path Length	Objective Value[15]	Time (Seconds)
0.99	0.01	640	145.43	1.4543	16544.90
0.85	0.15	638	133.77	21.7655	140982.00
0.82	0.18	633	109.45	25.4410	17433.2
0.64	0.36	620	85.94	31.8692	1954.26
0.44	0.56	613	80.38	56.8928	196.90
0.39	0.61	600	72.02	59.5322	104.17
0.38	0.62	592	67.11	59.8482	82.72
0.26	0.74	571	59.50	61.9700	2.92
0.24	0.76	552	53.20	62.2480	0.19
0.07	0.93	537	51.92	64.1792	0.02
0	1	537	51.92	51.9200	0.01

using MCSP and NR-MCSP models for the parameters described above (i.e. a series of maximum service distances and objective weights), the NR-MCSP model found better solutions in 527 of the 1600 problems that were solved. This represents approximately 33% of the total set of solutions. If we were to exclude maximum service distances in which the shortest path is able to cover all demands, the overall proportion in which loops/tours are utilized as an optimal covering strategy further increases. What is most significant in the results, however, is the fact that, as one shifts emphasis towards shorter overall path length, loops become more and more attractive as a covering strategy. This seems to validate the use of loops in designing real world transit routing applications. Moreover, solution times for the NR-MCSP performed similarly to the MCSP although in some cases the total solution times could take substantially longer

[15] Note that the objective is calculated by minimizing what is not covered plus distance

than the MCSP. These cases occurred where there were a number of sub-tours which were encountered and/or used in the solution process or in the final solution and were often associated with solutions where a greater emphasis was placed on coverage than distance. More time was also generally taken to solve problems with maximum service values that were between 7.5 and 15.

Table 3.4 – Unique solutions found for the MCSP using a maximum service distance of 10. The table includes coverage weights, distance weights, total path length, and path coverage for each unique solution.

Coverage Weight	Distance Weight	Total Path Coverage	Total Path Length	Objective Value[16]	Time (Seconds)
0.99	0.01	640	162.69	1.6269	21.53
0.84	0.16	638	151.83	25.9728	62.82
0.81	0.19	633	130.02	30.3738	43.13
0.69	0.31	618	95.71	44.8501	41.68
0.44	0.56	599	80.38	63.0528	23.94
0.39	0.61	586	72.02	64.9922	24.93
0.38	0.62	578	67.11	65.1682	19.25
0.26	0.74	557	59.50	65.6100	4.51
0.24	0.76	538	53.20	64.9120	1.87
0.07	0.93	523	51.92	56.4756	0.02
0	1	523	51.92	51.9200	0.01

The reason that some cases took longer to solve is that EAST constraints allow for a greater number of feasible solutions than traditional Dantzig, Fulkerson, and Johnson based tour breaking constraints. The time taken to resolve each unique solution, that is, the time taken to solve a model after insertion of EAST constraints, remained comparable to those of the original MCSP. Because the Dantzig, et. al. constraints utilized by Current, et. al. are much tighter, they effectively ensure that any form of a particular sub-tour cannot occur in a solution. In contrast, EAST constraints allow a tour to exist so long as arcs that were not part of the sub-tour are used to enter and leave the sub-tour.

[16] Note that the objective is calculated by minimizing a weighted combination of what is not covered plus distance

This means that the flexibility of the EAST constraints result in a phenomenon where the sub-tour will often ‘contort’ and ‘grow’ over time; as EAST constraints are identified and added to the problem the sub-tour has a greater set of possibilities that can potentially exist before finally being superseded by a solution in which that sub-tour attaches to the covering path. As an illustrated example, Figures 3.6 through 3.11 show a progression of intermediate solutions depicting the nature of a neighborhood of loops being generated for a problem where the maximal covering distance is 12.5. Figure 3.6 depicts a first solution where no EAST constraints are used, not even simple out-and-back (OAB) loop EAST constraints. This highlights the practicality of OAB loop EAST constraints, as these kinds of sub-tours are prevented from occurring and thus aids in reducing the number of times a sub-tour identification routine must be run. Solutions depicted in 3.7 to 3.11 depict intermediate solutions, each representing a subsequent solution generated after adding EAST constraints for each new sub-tour encountered in the previous solution. After repeating the sub-tour identification and constraint process, the final optimal solution is obtained which is shown in Figure 3.12. Another reason for which execution times can be longer is a result of the fact that there is a possibility of having more intricate sub-tours that are identified than in the original MCSP. These intricate sub-tours are combinations of sub-tours which are prevented at the outset when solving the original MCSP, but which might prevail in the new-revised formulation as they may lead to an enhanced solution. This also means that beyond a greater exploration of decision space, there may be more sub-tour identification and constraint addition iterations needed to completely converge to optimality.

In a similar vein, as the maximum service distance increases from 7.5 to 15, coverage provided by any node tends to increase, allowing for straighter, more direct covering paths, but also for short loops to be used to veer from a path to pick up additional node cover. Thus, the use of a loop in an optimal covering path problem can be frequent. At coverage distances below 5 there are very few nodes which can cover nodes other than themselves and for coverage distances greater than 15, most nodes are able to cover many other nodes which means that the variety of unique, efficient covering paths tends to decrease. However, the most significant contribution of this work lies in the fact that loops may be necessary when generating an optimal maximal covering shortest path solution, and that past MCSP models have been built upon an assumption that does not hold in general.

3.5 The Maximal Population Shortest Path Problem

The Maximum Population Shortest Path problem is a special case problem that involves a maximal covering distance of zero. This means that in order for a path to cover a node, that node must be on the path. Solutions to this problem were generated and reported in the previous section of the chapter, as they can be found using the NR-MCSP model where the maximum service distance is fixed at zero. Current et. al. even noted that MPSP solutions could be found using their original MCSP model where the maximum service distance is set to zero. However, they noted that a streamlined structure could be developed for the MPSP where fewer variable and constraints are needed as compared to what would be needed when solving an equivalent MCSP. Their streamlined model form for the MPSP uses the same notation that was introduced earlier

in this chapter. We can formulate the original MPSP model of Current et al. (1985) as follows:

MPSP1:

$$\text{Maximize} \quad Z_C - Z_D \tag{3.10}$$

Subject to:

$$\sum_{j \in F_p} x_{pj} = 1 \text{ for origin node } p \tag{3.11}$$

$$\sum_{i \in T_q} x_{iq} = 1 \text{ for destination node } q \tag{3.12}$$

$$\sum_{i \in T_k} x_{ik} - \sum_{j \in F_k} x_{kj} = 0, \forall\, k \in N \text{ where } k \neq p \text{ and } k \neq q \tag{3.13}$$

$$\sum_{i \in T_j} x_{ij} \leq 1, \forall\ j,\ j \neq p, q \tag{3.14}$$

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij} \leq |V| - 1, \forall \text{ subset of nodes } V \tag{3.15}$$

$$x_{ij} = (0,1), \forall (i,j) \tag{3.16}$$

Since the maximal coverage distance is zero, the path must visit node i for node i to be considered covered. The objective function is similar to the MCSP in the sense that it involves maximizing the population covered by the path (in this case the population at nodes visited by the path) while minimizing the resulting path length. Constraints (3.11), (3.12), and (3.13) are also exactly the same as in the MCSP. Constraint (3.14), however, requires that a node may be visited at most once. Thus, a path may or may not visit a node, but it cannot loop back or reverse course and visit that node again. This constraint will then ensure that the model has no loops as part of the covering path, as no more than one arc may be used to directly visit each intermediate node. However, utilizing such a constraint may require a more circuitous route than would be necessary if this requirement were relaxed by allowing a node to be visited a second time. Constraints

(3.15) are traditional Dantzig, Fulkerson, and Johnson sub-tour breaking constraints and constraints (3.16) are the necessary binary restrictions used to indicate arc selection for the path. It should be noted that constraints (3.15) are added to the formulation only when necessary. That is, the problem is solved without constraints (3.15), if any sub-tours are identified in the resulting solution, then selected constraints (3.15) are added to the formulation to prevent these sub-tours from occurring, and then the problem is re-solved. This process is repeated until no further sub-tours are identified as being part of the optimal solution. The above model does not contain a set of covering variables, as compared to the original MCSP model. By restricting the number of times that any node can be visited on the path to 1, allows one to use the arc variables as a proxy for covering whether a node is covered or not. Since the coverage distance is zero, and since a given node can be entered at most once along the path, the use of any arc to directly reach that node will represent whether that node is covered. Thus, we can define the coverage objective using only x_{ij} variables as follows:

$$Z_C = \alpha \sum_k \sum_{i \in T_k} a_k x_{ik} \qquad \textbf{(3.17)}$$

The fact that a path in the above model can visit a node at most once may lead to a path that is longer than necessary to cover a given level of population. That is, it is possible that some Pareto optimal solutions to the MPSP cannot be identified by model MPSP1.

To meet the stated objectives of the MPSP, that is, a node must be visited in order to provide service, and allow for the use of loops/tours as a covering strategy, the MPSP as formulated by Current et. al. (1985) cannot be modified to solve this less restricted problem (i.e. a formulation that allows loops). The MPSP formulation as stated by

Current et. al. (1985) is unsuitable to be amended to allow for the use of loops/tours as constraints (3.14) are used to ensure that a node's demands are considered covered if it is entered by an arc. However, since constraint (3.14) restricts a node to be only visited once by the path, this prevents a loop from being considered. Eliminating this condition will possibly lead to nodes being counted as covered more than in the covering objective (3.17). To eliminate any coverage double counting, one will need to re-introduce coverage variables and coverage constraints (3.5). In addition constraints (3.11) and (3.12) will need to be modified as well, as one cannot arbitrarily restrict that origin node as a point of exactly one departure or that the destination node is a point of only one arrival. Consequently all of the constraints except the integer restrictions and the constraints (3.13) will need to be modified to the forms found in NR-MCSP. Further, the sub-tour elimination constraints of (3.14) will need to be replaced by an EAST constraint set; essentially, all of these modifications lead to the model form of NR-MCSP given at the beginning of this chapter. That is, there is no special formulation that is possible for the MPSP problem when one allows loops to occur in a solution.

However, we can show that optimal solutions to the stated objective of the MPSP do, in fact, utilize loops/tours, if allowed, by setting the maximum service distance to zero in the NR-MCSP. Tables 3.1 and 3.2 present solutions to the NR-MCSP model and the MCSP model where the maximum service distance is set to zero and all of these solutions are optimal MPSP solutions. A comparison of two solutions, one to the MCSP and one to the NR-MCSP for this special case using a service distance of zero can be made by viewing the results depicted in Figures 3.13 and 3.14.

3.6 Concluding Remarks

In this chapter, we have presented a new, enhanced model for the maximal covering shortest path problem. We have shown that this model can be solved for modest size problem instances by off-the-shelf commercial software. We used the Xpress solver to generate over 1600 solutions to a range of problem instances. Solution times in many circumstances were comparable to the times needed to optimally solve the classic model of Current et al. (1985), even though this new model explores a larger solution space. The results have demonstrated that that embedded loops may be necessary in order to generate optimal covering path solutions. We have found that loops may be effective in several types of pathologies. One such case can occur when a loop is formed at the origin node or at the destination node. Other cases occur when there is a greater emphasis on coverage as compared to path length. This is particularly true when the maximum service distance is greater than zero. It is important to underscore the fact that the model formulated in this chapter, NR-MCSP, is loop agnostic, as it neither requires nor prevents loops from being used. This is a unique feature of the EAST constraints. Taken altogether we have seen that these model refinements yield solutions that are more efficient when loops are used. The MPSP model of Current et. al. (1985) forces intermediate nodes to be entered at most once and requires that the origin be left exactly once and the destination be entered exactly once. This will preclude the formation of a tour/loop in any solution. So, this too, is a model that may fail to find a true optimal solution whenever analyzing the distances for arcs used and coverage achieved. It is important to underscore that fact that whenever a loop is embedded in any optimal solution, all arc traverses are counted in terms of the distance travelled. So, if a route goes

out to one or several nodes and loops back on the same set of arcs, traversal costs in both directions are accounted for. In general, common intuition leads us to assume that such route/path strategies are non-optimal, and on the face of it is a poorly conceived approach. However, the results given in this chapter clearly demonstrate that true optimal solutions associated with the non-inferior tradeoff of coverage and distance may involve embedded loops. Therefore, we can see that the pioneering work of Current et. al. (1985) should be updated to include not only the use of EAST constraints as part of the sub-tour elimination process, but also allow for the possibility of a loop/tour to form at the origin and destination nodes.

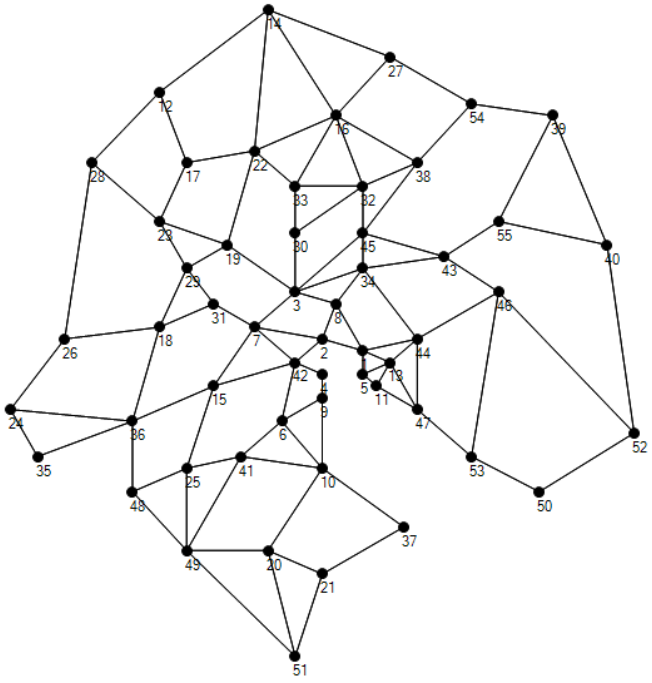


Figure 3.2 - The Swain based network. Nodes are labeled such that the higher population values are associated with lower node numbers and the lower population values are associated with higher node numbers. No distinction is made with respect to populations of the same value as these are assigned to the next sequential node number.

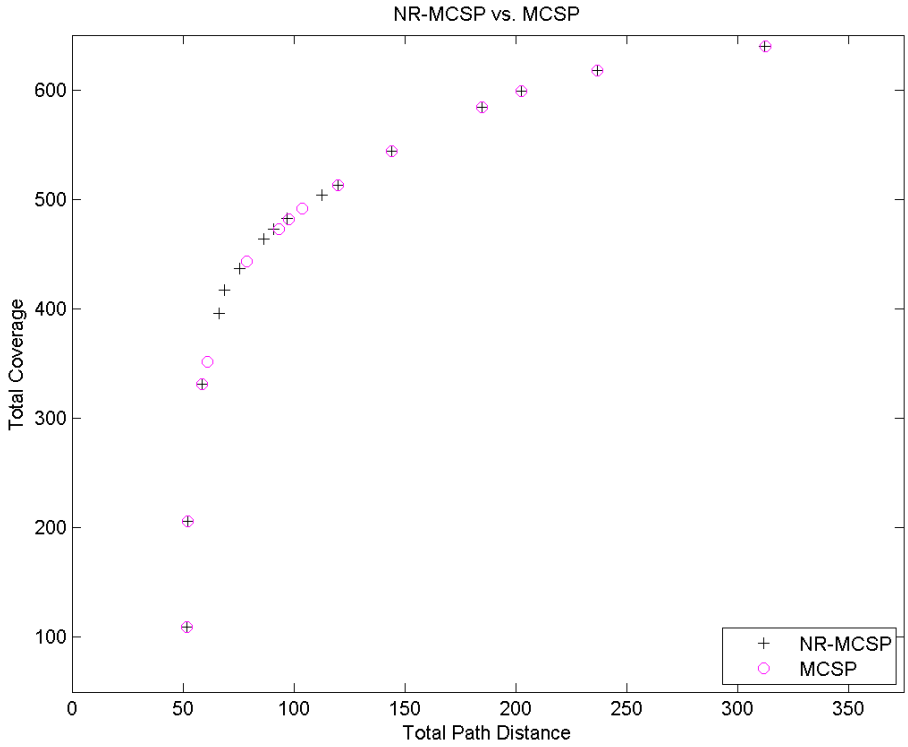


Figure 3.3 - The tradeoff curve with respect to coverage and distance for the MCSP and the new, revised MCSP models using a maximum service distance of *s* = 0

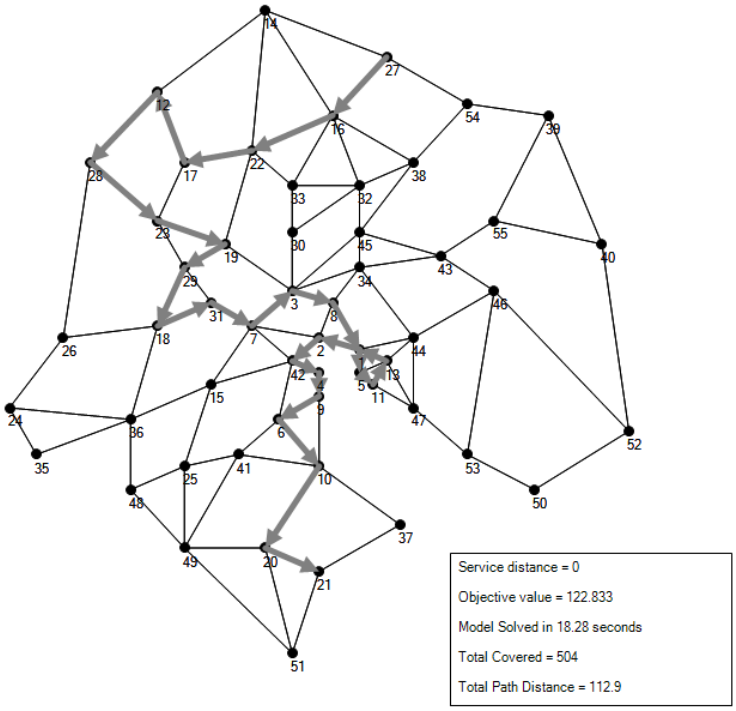


Figure 3.4 – An optimal NR-MCSP solution with a service value of 0 where the origin node is 27 and the destination node is 21. The coverage weight used was = 0.43, and the corresponding distance weight was = 0.57. Note that a loop/tour forms from node 1 to 5 to 11 to 13 and back to 1.

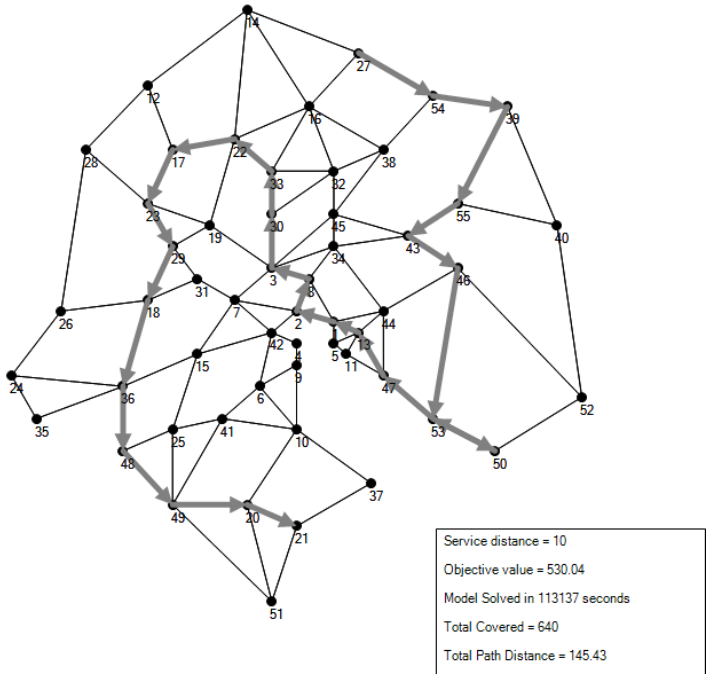


Figure 3.5 – An optimal NR-MCSP solution with a service distance of 10 where the origin node is 27 and the destination node is 21. This route corresponds to the solution generated when the distance weight = 0.14 and coverage weight = 0.86. Note the existence of loops/tours along several locations along on the path.

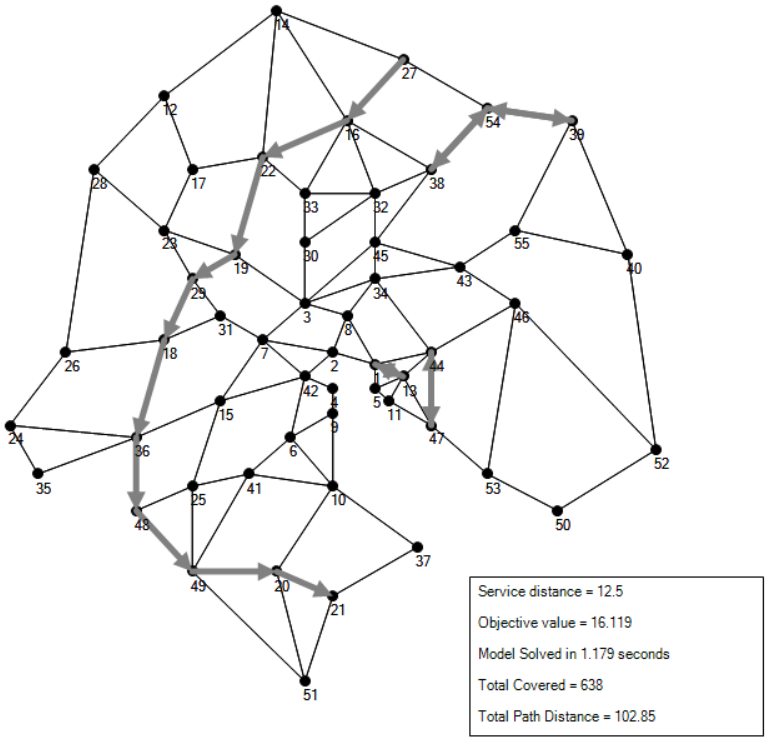


Figure 3.6 - Evolution of a sub-tour using EAST constraints 1

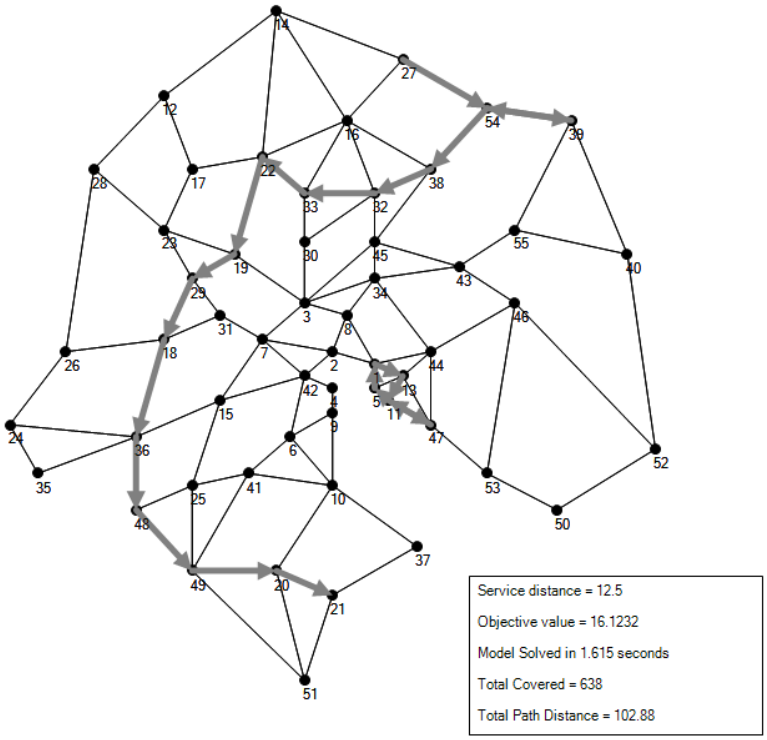


Figure 3.7 - Evolution of a sub-tour using EAST constraints 2

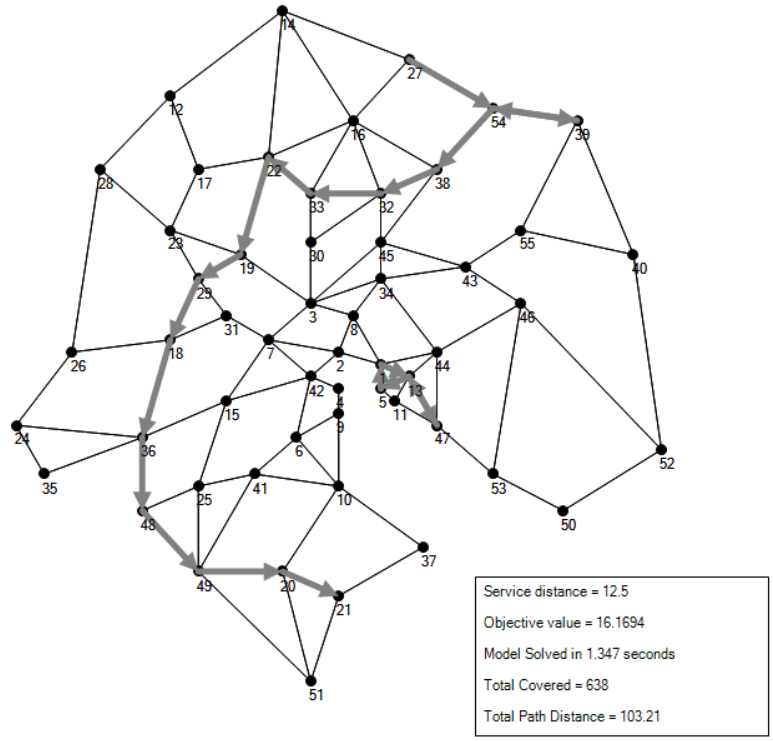


Figure 3.8 - Evolution of a sub-tour using EAST constraints 3

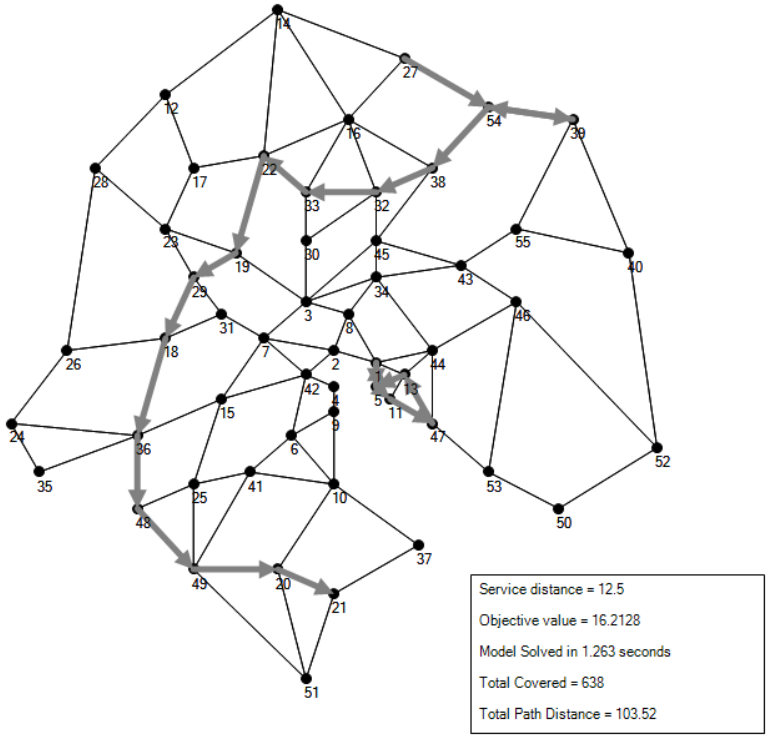


Figure 3.9 - Evolution of a sub-tour using EAST constraints 4

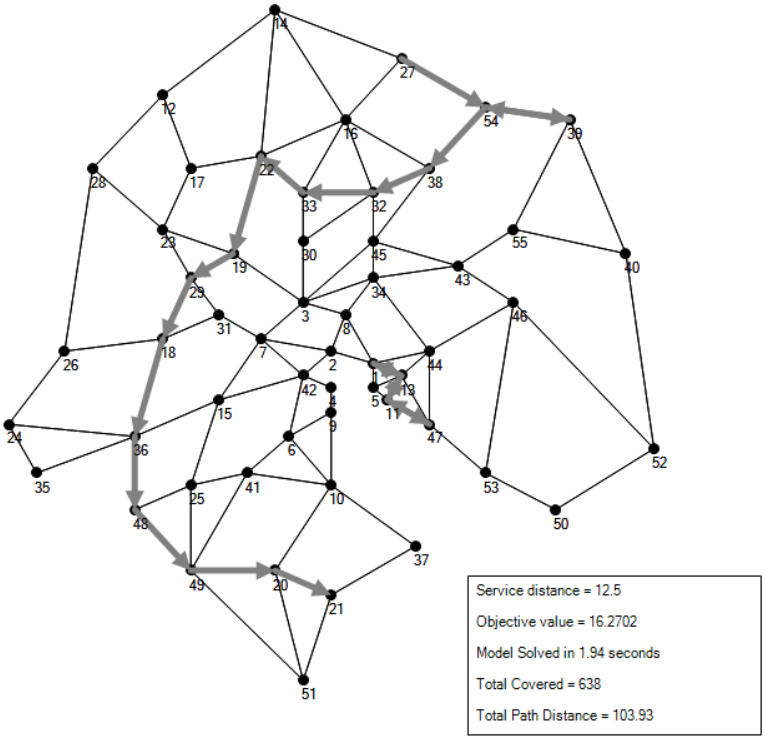


Figure 3.10 - Evolution of a sub-tour using EAST constraints 5

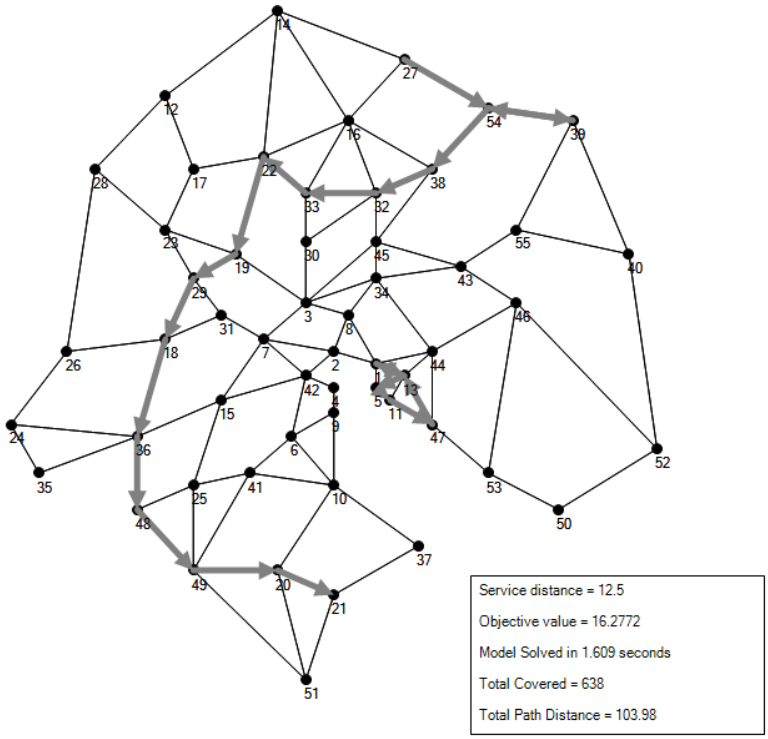


Figure 3.11 - Evolution of a sub-tour using EAST constraints 6

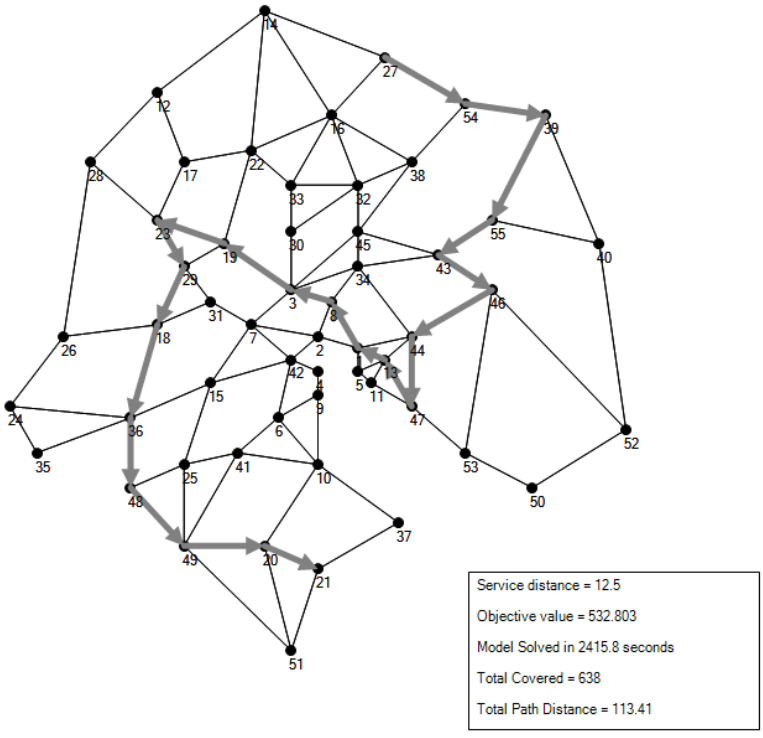


Figure 3.12 - An optimal NR-MCSP solution after the evolution of EAST constraints generated for a problem with a service distance of 12.5, an origin node of 27, and a destination node of 21. This route corresponds to the solution generated when the distance weight = 0.14 and coverage weight = 0.86.

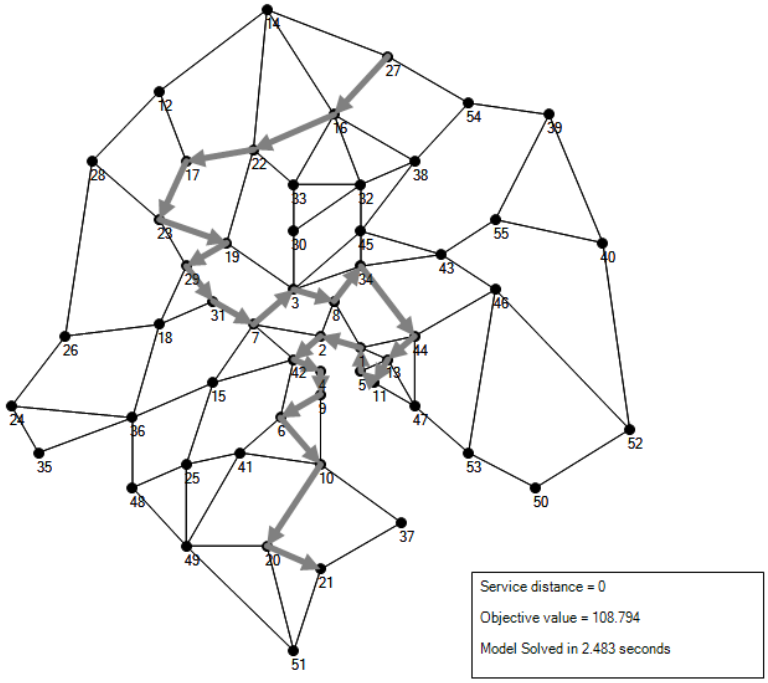


Figure 3.13 - Solution to the MPSP with a distance weight of 0.64 and a coverage weight of 0.36

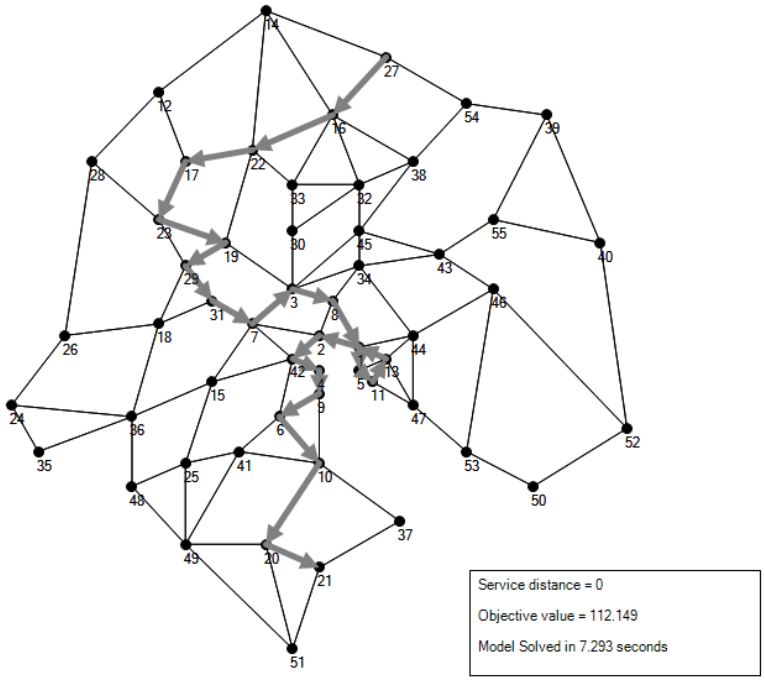


Figure 3.14 – An optimal NR-MPSP solution where the service distance is zero where the origin node is 27 and the destination node is 21. The distance weight was = 0.64 and the coverage weight was = 0.36.

Chapter 4

4.1 Introduction

The Maximal Covering Shortest Path problem (MCSP) is a multi-objective problem that involves maximizing weighted node coverage and minimizing path length. This problem is a combination of a path problem (solvable in polynomial time) and a maximal covering location problem that is classified as NP-hard. To say the least, there will be many problem instances of the MCSP that are unlikely to be solved to optimality using approaches such as linear programming combined with a branch and bound algorithm. Because there will be many problems that cannot be solved to optimality, it makes sense to develop heuristics for solving this problem. The objective of this chapter is to develop a flexible heuristic for solving the maximal covering shortest path problem that passively considers loops in seeking improvements.

Dufourd et al. (1996) were quite succinct in describing why they developed a heuristic for a covering-path problem. Their reasoning was as follows: "Rather than advocating the use of a large-scale mathematical model that would, in all likelihood, defy any known solution technique, we propose a practical and versatile heuristic approach for a simplified version of the problem…" (Dufourd et al., 1996). Because covering path and tour problems are classified as NP-hard, they are amongst a set of difficult to solve problems. Integer-linear programming, which was used in Chapter 3, is indeed limited in its ability to solve large problems to optimality. Because of this issue, Dufourd et al. (1996) and other have developed heuristics for this class of problems and it makes sense

to explore the development of the maximal covering-shortest path where loops are allowed.

As was noted in Chapter 2, heuristics are methods of deriving potentially good, but not necessarily optimal, solutions for a given problem. Heuristics have been applied on a broad range of NP-Hard problems. General heuristic solution methods include strategies such as: λ-opt (Lin, 1965); insertion (Gendreau, et al. 1998; Current and Schilling, 1994); greedy (Chvatal, 1979; Church, 1974); semi-greedy (Hart and Shogan, 1987); GRASP (Feo and Resende, 1995); ant colony optimization (Dorigo et al., 1999; Colorni et. al., 1991); genetic algorithms (Hosage and Goodchild, 1986); LaGrangian relaxation (Narula, 1977); heuristic concentration (Rosing and ReVelle, 1997); simulated annealing (Kirkpatrick et al. 1983; Golden and Skiscim, 1986); Tabu search (Glover, 1989; Rolland et al., 1997); variable neighborhood search (Mladenović and Hansen, 1997); threshold accepting (Dueck and Scheuer, 1990); the great deluge (Dueck, 1993), as well as many other methods.

Overall, there has been little attention paid to developing a heuristic with respect to the MCSP problem. Notable exceptions to this have involved the development of a heuristic for the Shortest Covering Path problem (Current et al., 1994); a heuristic for a transit route delineation problem defined on a grid (Dufourd, et al. 1996); a heuristic for a multi-route transit problem defined on a network that involved, routes, fleet size and headways (Fan and Machemehl, 2004), and a heuristic for a maximal covering route problem where each demand is served only when the route serves both their desired origin and desired destination (Fernandez and Marin, 2003).

The approach taken by Fernandez and Marin (2003) and Current et al. (1994) utilizes Lagrangian relaxation. The two formulations for these two related works both include explicit sub-tour elimination constraints that are related to Dantzig, Fulkerson, and Johnson's (Dantzig et al., 1954) earlier work on the TSP. With certain values of multipliers, it is possible that negative cost cycles will be present in network of the relaxed problem. In such a case, Current et al. (1994) ignore that particular set of multipliers. This allows them to use Dijkstra's (Dijkstra, 1959) shortest path algorithm in solving the relaxed problem. Dijkstra's algorithm does not admit cycles or loops in its process, which means that the resulting Lagrangian solution will not contain a loop. Fernandez and Marin (2003) take a different approach, by defining their problem on an acyclic network. This means that no solution could contain a loop or a cycle. This feature allowed them to drop the sub-tour elimination constraints and solve sub-problems using a shortest path algorithm. Fan and Machemehl (2004) utilized a genetic algorithm approach to solve their problem involving routes, headways, and fleet sizes. The initial population of routes were generated using the k^{th}-shortest path algorithm of Yen (1971). It should be noted that there are two types of k^{th}-shortest path algorithms: those that admit loops as a part of the path and those that don't. The Yen algorithm does not allow loops in the generation of paths. This means that overall, the Fan and Machemehl (2004) approach does not generate loops on a given route. The work of Dufourd et al. (1996) rests on the use of Tabu search.

The remainder of the chapter is outlined as follows. The next section will present an overview of the heuristic followed by a discussion of each step within the heuristic. The subsequent section will present results of the heuristic and compare them to known

optimal solutions. It will also discuss performance, and other computational issues. The gaps between known optimal solutions and the heuristic will also be presented. The final section will summarize the results and suggest future work.

4.2 The Maximal Covering Shortest Path Heuristic

The set of possible heuristic approaches can be considered to be large and diverse. As described in Chapter 2 it ranges from approaches that are inspired by biological mechanisms (Genetic algorithms & swarm smarts), chemical processes (Simulated annealing), simple greedy and semi greedy approaches, as well as simple perturbation mechanisms like insertion, swapping, and substitution. The work here is quite simple in its design and is meant to be used as a building block to future work. As most meta-heuristics, like Tabu search and simulated annealing, are built upon some form of solution perturbation that explores a neighborhood defined about the current solution, the design of the heuristic here is based upon a form of solution perturbation, which is based upon modifying the current incumbent solution by a simple change in route alignment. Changes in a route are based upon two techniques: a local insertion (which is a simple out and back loop being added to the current solution) and a detour substitution, which involves substituting a portion of the existing route alignment by a different alignment that is itself a local gateway path.

All heuristics are based upon a metric of solution value. Generally speaking heuristics tend to find those solutions which perform better in terms of this metric. Before we delve into the details of the heuristic, it is important to describe the form in which the metric of solution value is computed. To do this we define:

Z_C = the covering objective – the total coverage provided by the covering path **(4.1)**

Z_D = the path distance objective – the total length of the covering path **(4.2)**

The maximal covering shortest path problem is a multi-objective problem involving maximizing path coverage and minimizing path length. We can combine these two objectives 4.1 and 4.2 as the following composite objective 4.3:

$Z_{Obj} = a * Z_C - b * Z_D$ the composite objective associated with the covering path **(4.3)**

Where:

a = the importance weight associated with the covering objective (where $\alpha \geq 0$)

b = the importance weight associated with the path distance objective (where $\beta \geq 0$)

Whenever two solutions are compared, the solution with the higher composite objective value, Z_{Obj}, is preferred. In general, whenever a solution T is found that is better than the existing solution E in terms of Z_{Obj}, solution T becomes the new incumbent solution E.

Figure 4.1a depicts a hypothetical route between the designated origin and destination nodes. Figure 4.1b depicts a simple out and back loop route being attached to the existing route. Note that this out and back (OAB) loop is attached to a node designated as k. This OAB loop travels from node k to node q. This OAB loop takes the shortest path from node k to node q and then back again to node k. There are many possible out and back loops that could be used to modify and improve an existing route. Technically, there is one for every node k on the route times the number of nodes, q, that are not on the route. Some of these are nonsensical. Figure 2a depicts one of these nonsensical loops, which is formed by the route from 6→3→2→5→4→2→3→6. Such a loop when inserted would needlessly add to route length as compared to the OAB loop depicted in Figure 2b which simply goes from 2→5→4→2. Since these nonsensical OAB loops are clearly dominated by others they can be eliminated from further consideration.

As stated above the number of possible OAB loops equals K*(n-K) where K is the total number of nodes on the current path and n is the number of nodes on the network, which means that as network size increases, the number of possible OAB loops also increases. This is one of the reasons that a good heuristic is desirable in solving this problem.

Figure 4.3a depicts a route alignment connecting the origin and destination nodes. It also depicts two chosen nodes along the route, u and v. For any given nodes u and v, there exists a gateway shortest path from u to v via node q. Figure 4.3b depicts a gateway path starting at node u which goes to the gateway node q and then continues to node v. The essence of the heuristic is to utilize a detour substitution by taking the existing route from the origin, O, to the destination, D, and consider changing the route from O to D, by going from O to u along the existing route, and then from u to q to v along the gateway path, and then complete the route from node v to node D along the existing route alignment. If a gateway path can substitute for a portion of the existing route and improve the composite objective, then this modified route becomes the new solution E.

The general form of the new maximal covering shortest path heuristic we have derived is given below. We assume there is a planar network of arcs and nodes such that each node has an associated demand and cover can be provided through the connected arcs of the network to all other demands based upon a maximum service distance from a node that is visited by the path. We base our heuristic on finding an initial shortest path and then begin an insertion process that greedily chooses the best insertion it can make. The generalized form and notation is as follows:

O = origin node
D = destination node

d_{ij} = the shortest distance through the network from node *i* to node *j*

s = the maximum distance that service can be provided

c_j = the set of nodes *I*, such that $d_{ij} \leq s$, that can be covered if node *j* is visited on the covering path

a = the emphasis associated with the covering objective

b = the emphasis associated with the total path distance objective

Z_C = the covering objective – the total coverage provided by the covering path

Z_D = the total path distance objective – the total length of the covering path

$Z_{Obj} = a * Z_C - b * Z_D$ the objective value associated with the covering path

e = the maximum number of possible iterations the heuristic can perform

ℓ = the current iteration number

G = the set of nodes which are not directly visited by the covering path indexed by ascending order (e.g. $f = \{1, 3, 4, 5, 7, ..., n\}$)

g = a node within the set G that is to be swapped and which is not a part of the current covering path

f = the set of nodes that are currently a part of the covering path but which are not equal to *O* or *D* and which is indexed in ascending order (e.g. $f = \{1, 3, 4, 5, 7, ..., n\}$)

P_i = the point of departure associated with travel from the origin node to a node *i*

P_j = the point of entry associated with travel from a node *j* to the destination

CP_{Best} = the set of arcs for the current best maximal covering shortest path that has been found

$CP_{Candidate}$ = the set of arcs for the current maximal covering shortest path candidate

$CP_{Current}$ = the set of arcs which comprise the current maximal covering shortest path

CP_{P_i} = the set of arcs which are used to go between the origin, *O*, and P_i on the current covering path

CP_{P_j} = the set of arcs which are used to go between P_j and the destination, *D*, on the current covering path

CP_g = the set of arcs which are used to go from P_i to *g* and *g* to P_j

Step 1: Define the parameters of the heuristic – set the maximum service distance, *s*, and select an origin node, *O*, and destination node, *D*, for the covering path. Set the desired coverage and distance emphases weights such that each weight value lies between 0 and 1 and $a = 1 - b$. Set the maximum number of iterations, e, that can be used should the heuristic not converge on a solution. Go to *Step 2*.

Step 2: Determine the shortest path through the network from *O* to *D*. Calculate $Z_{Obj}(CP_{Current})$ and save this solution as the current

maximal covering shortest path, and set $CP_{Current}$ to be the set of arcs that comprise the shortest path from *O* to *D*. Set $Z_{Obj}(CP_{Best}) = Z_{Obj}(CP_{Current})$ and $\ell = 0$ and go to *Step 3*.

Step 3: Set $\ell = \ell + 1$ Go to *Step 4*.

Step 4: If *Step 4* has been reached from *Step 3* then set P_i equal to the first element (node) in f. Else, set P_i equal to the next element in f. Go to *Step 5*.

Step 5: If *Step 5* has been reached from *Step 4* then set P_j equal to the first element in f (note that P_j can be the same node as P_i). Else, set P_j equal to the next element in f. Calculate the respective path elements for CP_{P_i} and CP_{P_j}.

Step 6: If *Step 6* has been reached from *Step 5* set *g* to be the first element in *G*; else, set *g* to be the next element in the set. Calculate CP_g and set $CP_{Candidate} = CP_{P_i} + CP_g + CP_{P_j}$. Go to *Step 7*.

Step 7: If $Z_{Obj}(CP_{Candidate}) \geq Z_{Obj}(CP_{Best})$ then set $CP_{Best} = CP_{Candidate}$. If *g* does not equal the last element in the set *G* then go to *Step 6*; else go to *Step 8*.

Step 8: If P_j does not equal the last element in f go to *Step 5*; else if P_j equals the last element in f and P_i does not equal the last element in f then go to *Step 4*; else if P_j and P_i equal the last element in f check whether $Z_{Obj}(CP_{Current}) = Z_{Obj}(CP_{Best})$; if this is true then stop, the solution has not changed. Otherwise, set $CP_{Current} = CP_{Best}$; if $\ell = e$ then stop; else update *G* and f with respect to $CP_{Current}$ and go to *Step 3*.

Now that we have seen the basic structure of the heuristic, each step can be explained in further detail. *Step 1* in the maximal covering heuristic above sets all the parameters required to start the heuristic. Since we are trying to find good solutions to the MCSP we need to define the origin and destination nodes for the path as well as the maximum service distance which defines the maximum distance at which a node visited

on the path can provide coverage to other nodes. We also need to define how much emphasis is placed on coverage versus total path length. The last element is a stopping criteria used to set the maximum number of iterations the heuristic can run through before exiting. Ideally this number would be relatively large in proportion to the problem size. It is used to ensure that the heuristic has a stopping criterion should a solution not be converged upon within a reasonable amount of time.

Step 2 is used to define an initial covering path from which the heuristic can start. In this case we simply find the shortest path from origin to destination and set this to be the current covering path. We can now set the iteration count at the initial value of zero and set the best objective found to be that corresponding to the shortest path from *O* to *D*. The reason for setting the best objective to that defined by the shortest path is to establish a baseline objective so that better solutions can be initially identified. Since the shortest path represents a lower bound on the optimal solution, we can use it to initialize the heuristic with relatively little computational effort expended.

Step 3 begins the search procedure for the heuristic. In this case we now set the iteration counter to be one and we begin to move through the steps of the heuristic search process. The iteration counter allows us to keep track of the best solutions determined for each search iteration. The next step after initializing and updating the iteration counter is *Step 4*; If *Step 4* has been reached directly from *Step 3* then we select the first element in the set f. The first element in f should correspond to the node that has the lowest index number of the set of nodes that currently comprise the covering path. For example, if nodes 3, 5, 8, 18, 27, and 31 lie on the covering path, the first element in the set would be node 3. If *Step 4* is not directly reached from *Step 3* then the next element in the set f is

selected. Using the example set given above, if *Step 4* has been entered three times, the node represented at element position 3 should be selected – in this case that corresponds to node 8. Thus, *Step 4* allows us to select a 'departure point' utilizing the current maximal covering shortest path as a basis. We will then use this basis to compile a test-covering path. The departure point defined in *Step 4* will be the position along the covering path where a test insertion will originate as part of the test-covering path. As such, we set P_i, the 'departure point,' to be equal to the selected element in f which corresponds to the node on the path where the test insertion will originate. Once this has been completed we move to *Step 5*.

In *Step 5* we want to establish an 'entry point' where the test insertion enters the existing covering path. The first portion of *Step 5* is similar to that of *Step 4*. If *Step 5* is entered directly from *Step 4* then we select the first element in f; otherwise we want to select the next element in the set. The node represented by that element will then be set as the 'entry point,' P_j, for the test-covering path. It should be noted that in this case both the 'entry point' P_j and the 'departure point' P_i can correspond to the same node. By allowing both the entry point and the departure point to be the same we allow an OAB tour to be inserted onto the test-covering path. The next portion of *Step 5* determines the path elements that are used to define the test-covering path. At this point, the heuristic determines the set of arcs which are used to travel from the origin node to the point of departure on the current covering path as CP_{P_i}. The heuristic also calculates the set of arcs on the current covering path which are used to travel from the point of entry to the destination node as CP_{P_j}. Together this will allow the heuristic to keep track of which

portions of the current covering path that are kept, and which portions of the current covering path that will be removed and replaced with a newly defined insertion.

Step 6 determines what the test-covering path will be composed of. In this case we want to first select a node which is not currently part of the covering path. If this is the first time *Step 6* has been entered from *Step 5* then we select the first element of the set *G*, and define this node as *g*. If *Step 6* has not been directly entered from *Step 5* then we select the next element in the set *G*. We also need to define the path from the 'departure point' P_i, to *g* and from *g* to the 'entry point' P_j. To do this we find the shortest path from P_i to *g* and the shortest path from *g* to P_j. We define this insertion as CP_g. Once we have determined the associated arc set that defines the insertion we can define the test-covering path as $CP_{Candidate}$ which represents the newly defined candidate path composed of CP_{P_i}, CP_{P_j}, and CP_g. Once we have established the candidate covering path we move to *Step 7.*

In *Step 7* we calculate the objective value for the candidate covering path. Since we have a set of candidate arcs we are able to determine which nodes on the network are covered by the candidate path and can calculate the associated coverage. We also know the distance associated with each arc and since we know the order of arcs we are able to determine the total length of the path. Therefore, we can determine the composite objective associated with the candidate covering path; in this case weighted total coverage minus weighted total distance. This allows us to make a direct comparison with the current best covering path that has been determined. If the objective for the candidate covering path exceeds the current best covering path then the current best covering path is set to be equal to the current candidate covering path. If *g* is not the last element in the

set *G* then the heuristic returns to *Step 6* and the process is repeated; otherwise if *g* is the last element in the set *G* then the heuristic proceeds to *Step 8*.

In *Step 8* of the heuristic process we determine if all possible candidate covering paths have been checked with respect to 'departure points' and 'entry points' on the current covering path. If the entry point, P_j , is not the last element in the set f then the heuristic moves to *Step 5* where the next entry point can be selected from the set and the heuristic continues. If the entry point, P_j , is the last element in the set f and the 'departure point,' P_i, is not the last element in f then the heuristic must move to the next 'departure point' in the set f. Thus, the heuristic must go to *Step 4*. If both the 'departure point' and the 'entry point' are the last elements in the set f then the heuristic checks to see if an improvement to the current covering path has been found. Thus, the first part of *Step 8* is used to determine which step the heuristic should return to.

If no better solution has been found for a complete iteration cycle – that is, cycling through the set f for all departure and entry points – then the heuristic stops. Otherwise, the heuristic sets the current covering path to be equal to the current best found covering path. If the heuristic has reached the stopping criterion for the maximum number of iterations the heuristic stops. Otherwise, sets *G* and f are updated to reflect the new basis – i.e. the new current covering path that was just set – and the heuristic returns to *Step 3* and the process is repeated. Thus, the heuristic is able to quickly determine good solutions through a logical process. In this case the steps of the heuristic outlined above test a variety of insertions based upon an initial shortest path in a desire to determine a good (and hopefully optimal) solution to the maximal covering shortest path problem. Although this method does not guarantee optimal results, it is a logical method

and our experience shows that we are able to determine a robust set of good solutions. As such, the following section will discuss issues related to the performance of the heuristic as applied to a test network.

4.3 Computational Experience

This section highlights the computational experience of the Maximal Covering Shortest Path heuristic defined in section 2. Particular attention is given to the time taken to determine a solution using the heuristic as well as the time taken to determine an optimal solution to the NR-MCSP using the same parameters. We highlight a set of solutions where we give the associated weights used to emphasize coverage and shortest overall path length as well as the maximum service distance used to define coverage within the network. We also highlight the total coverage provided for both the heuristic and optimal solutions as well as their associated total path distance and provide the composite objectives for both the heuristic solution and the optimal solution for each problem and the associated solution gap, if any. Solutions derived from the New, Revised Maximal Covering Shortest Path problem are indicated with an NR in the problem heading and solutions determined from the heuristic are indicated with a H. Solutions in which a loop/tour are used as a covering strategy are marked with an asterisk, *, next to the problem type in the 'Problem' column in Table 4.1 which highlights these findings below.

Table 4.1

Problem	Maximum Service	Distance Weight	Cover Weight	Solution Time (s)	Composite Objective	Total Cover	Path Length	Gap Abs	%
NR	10	0.01	0.99	16544.9	632.15	640	145.43	--	--
H*	10	0.01	0.99	0.68	632.08	640	151.64	0.065	0.01
NR	10	0.15	0.85	140982	522.23	638	133.77	--	--
H*	10	0.15	0.85	0.49	521.30	638	139.98	0.930	0.2
NR*	10	0.19	0.81	4641.39	491.94	633	109.45	--	--
H*	10	0.19	0.81	0.31	490.28	633	118.17	1.660	0.3
NR*	10	0.3	0.7	219.76	410.27	633	109.45	--	--
H*	10	0.3	0.7	0.28	408.92	633	113.92	1.350	0.3
NR	10	0.56	0.44	196.90	224.71	613	80.38	--	--
H	10	0.56	0.44	0.15	224.71	613	80.38	0	0
NR	10	0.61	0.39	104.17	190.07	600	72.02	--	--
H	10	0.61	0.39	0.15	190.07	600	72.02	0	0
NR	10	0.62	0.38	82.72	183.35	592	67.11	--	--
H	10	0.62	0.38	0.09	183.35	592	67.11	0	0
NR	10	0.74	0.26	2.92	104.43	571	59.50	--	--
H	10	0.74	0.26	0.07	104.43	571	59.50	0	0
NR	10	0.76	0.24	0.19	92.05	552	53.20	--	--
H	10	0.76	0.24	0.07	92.05	552	53.20	0	0
NR	10	0.93	0.07	0.02	-10.70	537	51.92	--	--
H	10	0.93	0.07	0.04	-10.70	537	51.92	0	0
NR	10	1.0	0.0	0.01	-51.92	537	51.92	--	--
H	10	1.0	0.0	0.04	-51.92	537	51.92	0	0

It can be seen that the newly devised heuristic for this problem solves with times that never exceed one second – in fact the longest amount of time the heuristic took to

solve was just over two-thirds of a second at 0.68 seconds for a coverage weight of 0.99 and a distance weight of 0.01. Solution performance of the heuristic was also quite good with the percent gap between the heuristic solution and the optimal solution never exceeding one half of one percent. This gap could likely be further reduced through implementation of a GRASP (Greedy Randomized Adaptive Search Procedure; Feo and Resende, 1995) implementation as this would allow perturbations which could potentially flush out the optimal solution. However, although this heuristic uses a best insertion approach, results that we obtained are robust and computational time is significantly lower than if one solves the problem to optimality.

Table 4.1 also shows that the population that was covered in heuristically determined solutions was equivalent to the total population covered by the optimal solution. Although this result could change with respect to network geometry (i.e., a grid based network, vs a hexagonal network, vs a random network, etc.) it seems that significant changes are unlikely to occur as coverage is defined as being within a maximum service distance which is a function of network distance. What is of particular note is that the heuristic determined solutions which contained loops in several solutions. Although some of these 'looped' solutions are not globally optimal the gap between the optimal solution and those derived by the heuristic are quite low. This seems to lend credence to the fact that planners can consider using loops and not necessarily feel that they have compromised on what's best when deriving bus routes within urban and suburban areas.

Table 4.2 shows results for both the heuristic as well as the NR-MCSP as applied to the Garland/Richardson, TX network such defined by Curtin and Biba (2011). We

have chosen to use this network as it is representative of what would likely be used in a transportation planning context. Curtin and Biba note that the network is defined by major arterial roads which are likely to have ample room for busses to maneuver. To add data to the network, we were able to take the network defined by Curtin and Biba and utilize Open Street Map (OSM) data to define the complete road network. We were then able to crawl Zillow.com using cyber search methods and obtain the parcel information for Garland/Richardson, TX. We then defined the centroid for each parcel and connected the centroid to the nearest underlying road. Each centroid then represents one household. These households allowed us to define a 'population' (in this case households served) at each intersection node. Each centroid was then assigned to the nearest node of the Curtin and Biba network using the ArcGIS network analyst toolbox. We were then able to define the number of households served by each node based upon how many of these attached centroids were within 660 ft (one-eighth of a mile) using the underlying road network. A similar method was proposed by Biba et. al. (2010). Once the data was processed, we created a data file that can be used as an input to the NR-MCSP problem that can be solved by the Xpress solver as well as be used for the program that uses the heuristic that is defined in this chapter. It should be noted that when solving the problem, we utilized a maximum service distance that was equal to zero. Since we defined how many households were within an eighth of a mile when establishing the number of households that constituted the population for each node, it would be improper to use a service distance in the model and heuristic as this has already been accounted for.

Table 4.2 – The Comparison of the Heuristic and the NR-MCSP model as applied to the Richardson, TX dataset. An origin node = 70 and a destination node = 79 was used for all instances. A star (*) next to the problem name indicates that a loop is used.

Problem	Max Service	Distance Weight	Cover Weight	Solution Time (s)	Composite Objective	Total Cover	Path Length	Gap Abs	Gap %
NR	0	0.99	0.01	0.3	-151.8	164	155	--	--
H	0	0.99	0.01	0.1	-151.8	164	155	0	0
NR	0	0.85	0.15	0.4	-107.2	164	155	--	--
H	0	0.85	0.15	0.1	-107.2	164	155	0	0
NR	0	0.70	0.30	0.5	-35.3	454	245	--	--
H	0	0.70	0.30	0.1	-59.3	164	155	24	67
NR*	0	0.55	0.45	8198.7	114.3	1283	842	--	--
H	0	0.55	0.45	1.48	102.2	843	504	12.1	10
NR*	0	0.40	0.60	171326.0	461.2	1706	1406	--	--
H*	0	0.40	0.60	46.2	450.0	1800	1575	11.2	2.4

Just as in the results shown in Table 4.1, all NR-MCSP calculations were performed on the Ubuntu 14.04 LTS server defined above. Although we generated a series of solutions with emphasis weights for distance and coverage ranging from 0.01 to 0.99 incremented in intervals of 0.01, only several solutions are shown here for comparative purposes. Table 4.2 only has several entries as at the time of writing the solver was not able to determine a complete set of solutions. However, the heuristic performs very well in the cases that were tested. Only one solution had a result that was far from optimal. Apart from this exception, in all the other cases that were tested the heuristic was always within 10% of the optimal solution. All solutions also utilized loops with the exception of those solutions that were the shortest path.

The use of loops appears to be particularly useful as emphasis is placed on the covering objective. Thus, it seems the challenging factor in this case lies in determining a least cost path which covers as much as possible. A loop then seems to be a useful tool to provide good coverage while also not greatly increasing overall path length. It is also possible that a GRASP based approach would potentially determine better, hopefully optimal, solutions. These may or may not involve solutions which involve a 'loop' as a perturbation by periodically forcing the heuristic process to utilize a randomly determined solution which does not initially improve the objective may in the end allow the heuristic to avoid local optima. In any case, a GRASP approach utilizing the basic heuristic procedure defined above is certainly an avenue that will be pursued in future work.

Nevertheless, the heuristic we define in this chapter is able to determine optimal solutions as well as solutions that cover as much as the optimal solution albeit with a slightly longer path. Another intriguing aspect of the heuristic performance is that it, too, is loop agnostic in the sense that loops/attached tours are neither explicitly prevented nor encouraged to form. In fact, in several near optimal solutions that were determined, the heuristic utilized a loop/tour as part of the solution. Such a case is shown in Figure 4.4 below. In this case a loop is utilized between node 43 to node 55. As stated above, this does seem to indicate that the use of a loop can be a practical strategy. Figure 4.5 is another case where a loop structure is used in the heuristic determined solution. In this case several loops are employed.

4.4 Summary

This chapter has introduced a new heuristic that is "loop agnostic" – that is it neither requires nor prohibits loops from forming in a solution – and which performs computationally well. The heuristic methodology was highlighted such that each step is presented in detail. The heuristic uses an insertion strategy such that if a better insertion is found it is taken and the process is repeated until no further improvements can be made. Computational experience indicates that the heuristic is able to cover much of the demand although it does not always determine the optimal maximal covering shortest path in several cases. However, the gap from the heuristic solution to the optimal solution is consistently less than half a percent with the maximum gap in any solution being 0.3 percent. For the Swain based dataset. The Richardson, TX based dataset shows that the heuristic is able to determine robust solutions, though it does not always determine the optimal solution. Additional strategies for modifying the heuristic include a GRASP based approach which would allow for a randomized insertion (one of the best insertions rather the best insertion candidate). This would help to expand the region of exploration.

Figure 4.1 - An example of a loop

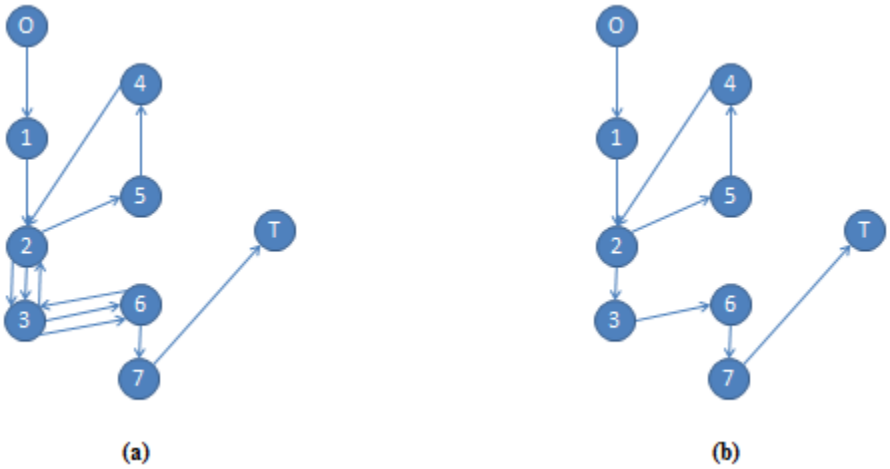


Figure 4.2 - A nonsensical and realistic example of a looped route

Figure 4.3 - A path from O to T and associated changes using gateway paths from u to q and q to v

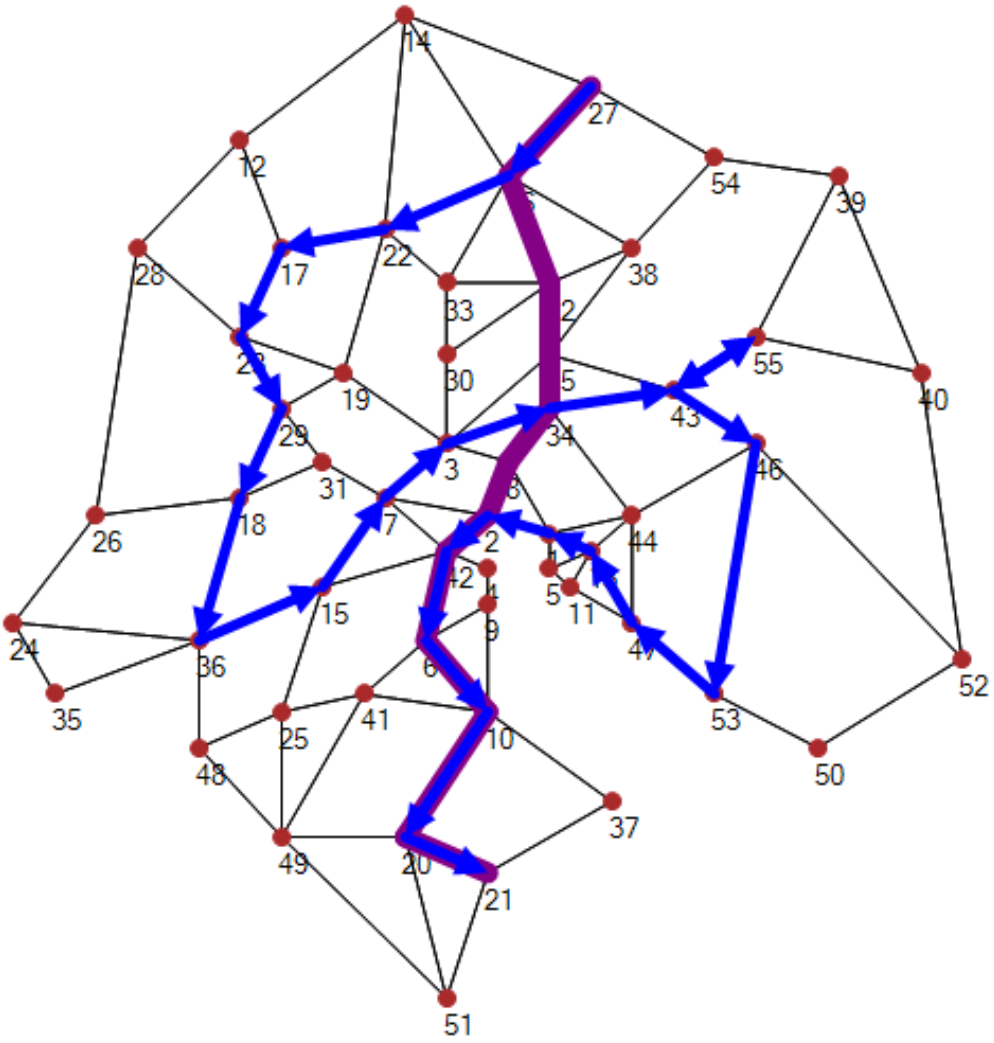


Figure 4.4 - Solution to the MCSP Heuristic determining a solution from node 27 to node 21 using a coverage weight of 0.85, a distance weight of 0.15, and a maximum service distance of 10. The mauve path is the initial shortest path and the blue path represents the final solution determined by the heuristic.

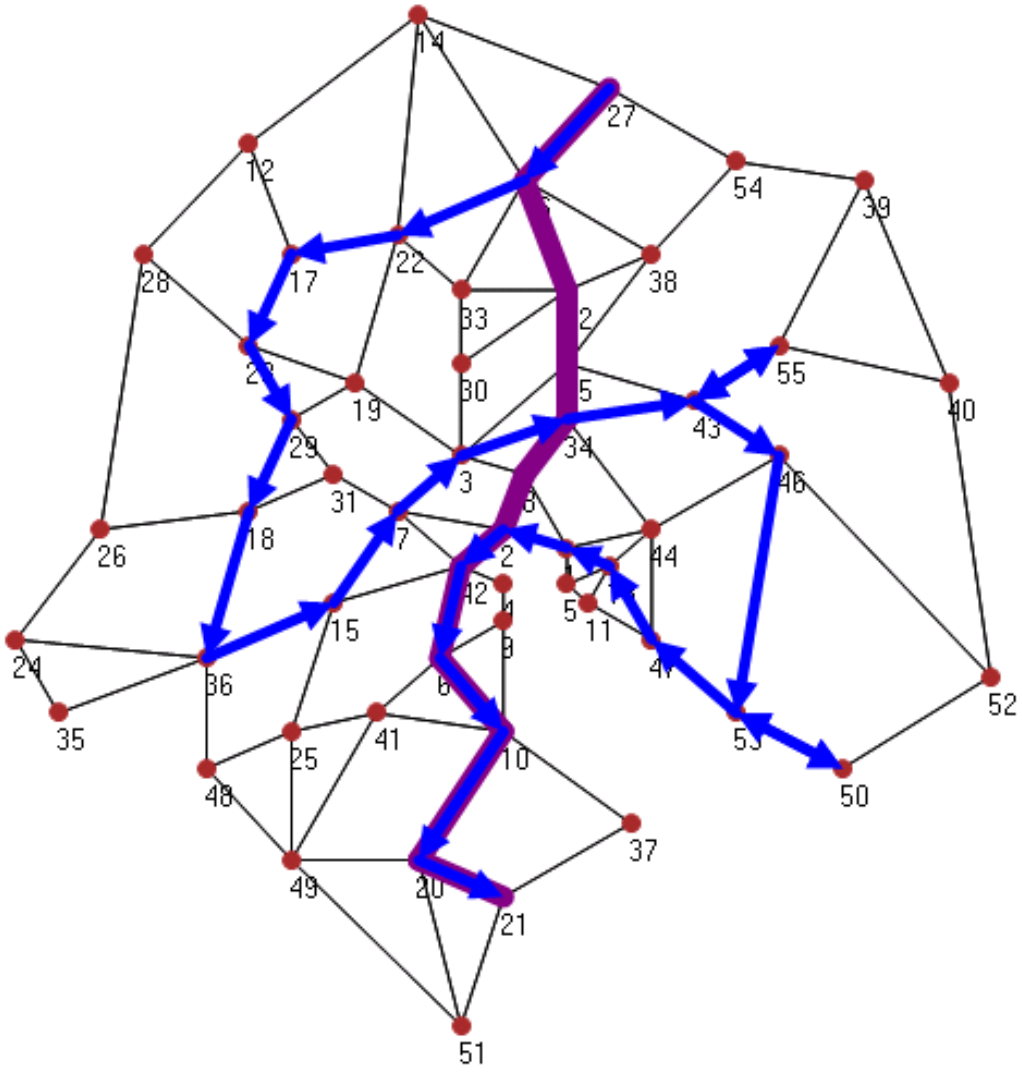


Figure 4.5 - Solution to the MCSP Heuristic determining a solution from node 27 to node 21 using a coverage weight of 0.90, a distance weight of 0.10, and a maximum service distance of 10. The mauve path is the initial shortest path and the blue path represents the final solution determined by the heuristic.

Chapter 5

5.1 Introduction

The goal of this dissertation is to develop new and improved models for the maximal covering shortest path problem. In chapter 3, a new model called the New, Revised Maximal Covering Shortest Path model was presented. This model was termed loop agnostic, as it neither requires nor prevents loops from being used in covering paths. Upon reflection, one can question whether a "loop" or "cycle" can be present if it is defined as a path. Some might suggest that this type of problem should be called the maximal covering shortest route problem instead. However, others have defined cases where paths can contain cycles or loops, especially with regard to the k^{th} shortest path problem. For example, the literature on the k^{th} shortest paths problem is bifurcated between allowing loops in such paths and not allowing loops. Quibbling about such terminology itself is not central to the research here. The issue here is to restructure models so that they can use loops or cycles, if the use of a loop or cycle leads to a solution which dominates one in which a loop or cycle is not permitted. As was shown in Chapter 3, where the NR-MCSP model is described and solved, a loop or cycle can be an optimal construct in a covering path. Chapter 4 presented a basic swapping heuristic for solving the NR-MCSP. Its application demonstrated that many close to non-inferior solutions exist that involve one or more loops. This also gives credence to current planning practice in transit operations where planners often design routes/paths with loops.

Up to this point we have examined the models originally developed by Current et. al. (1984, 1985) or derivatives of these models. Their original problem involved finding the shortest route through a network from a prespecified origin to a prespecified destination such that all nodes were covered by the path. A node j is defined as covered as long as the shortest covering path passes through node j or passes through some other node that is within a maximal service distance S of node j. This original problem was later expanded into the Maximal Covering Shortest Path problem. This newer problem is based on relaxing the requirement of complete coverage and instead involves the search for those paths that minimize path length and maximize path coverage. No MCSP solution needs to be longer than the shortest covering path. As was described in Chapter 2, covering-path type problems are related to other forms of path and routing problems such as the K^{th} shortest paths (KSP) problem and the Traveling Salesman Problem. Because of this, many of the models and even the solution procedures have been based upon the TSP and KSP literature.

Current et. al. (1984, 1985) developed integer-programming models for both the SCP and MCSP. These two models represented a combination of the shortest path model of Dantzig (1956) with TSP sub-tour elimination constraints developed by Dantzig, Fulkerson, and Johnson (1954). Coverage constraints were added to either enforce coverage (as in the location set covering problem) or defined if a specific node is covered (as in the Maximal Covering Location Problem). However, other methods have been devised to solve the Traveling Salesman Problem through alternate formulations which do not require the use of sub-tour elimination constraints as defined by Dantzig, Fulkerson, and Johnson. Such formulations include the work of Miller, Tucker, and

Zemlin (1960), Vajda (1961), and Gavish and Graves (1978).[17] Although there have been many formulations of the Traveling Salesman Problem that have been developed and tested, virtually all work with respect to covering path problems have utilized the solution procedure and constraints originally developed by Dantzig, et al (1954) and adapted by Current et al (1984, 1985) for use in their seminal formulations. There are two exceptions to this general trend: the EAST constraints of Niblett and Church (2016) and the TRANSMax model of Curtin and Biba (2011).

The Transit Arc Node Service Maximization (TRANSMax) model was developed by Curtin and Biba (2011) which is adapted from an alternative model for the Traveling Salesman Problem that was developed by Vajda (1961). The TRANSMax model can be considered a special case of a maximal covering path problem where arcs on the path provide coverage to nodes as well as to demand defined to exist along arcs. In essence, it is an alternate model formulation for the MCSP. The TRANSMax model was discussed in detail in the literature review of Chapter 2. If we recall from Chapter 2, the routing construct of Vajda (1961) expressively prevents a travelling salesman tour from crossing itself as well as looping back. The Vajda formulation is constructed in such a way as to eliminate the need for sub-tour elimination constraints like those proposed by Dantzig, Fulkerson, and Johnson. By expanding on Vajda's formulation, Curtin and Biba (2011) made it possible to solve for a path that maximizes node and arc coverage while restricting the path to be no longer than desired. The Vajda construct expressively prevents a path or route that 'loops back' upon itself as well as ensures that the path or route is connected. In light of the work and results of Chapter 3, it makes sense to

[17] Orman and Williams (2004) give an excellent review of these problems and the efficacies of each formulation with respect to solution time and general computational performance.

explore the possibility of transforming the TRANSMax model into an approach which is 'loop agnostic' – that is, a formulation in which the use of a loop is allowed if it represents an improvement in the objective function. This is the goal of this chapter.

This chapter is organized as follows; we start with a brief review of the original TRANSMax model and follow that with a revised form that allows for out-and-back loops and cycles. We also describe and discuss the key differences between the revised model – TRANSMax II – and the original TRANSMax formulation, particularly with respect to being 'loop agnostic.' Both the original and new TRANSMax models are applied to data that represents Richardson and Garland, Texas which was also used in Chapter 4 when testing the performance of the heuristic. The choice of this network follows the work of Curtin and Biba (2011). We will show that optimal solutions to the revised TRANSMax problem mirror those of the NR-MCSP for this real world road network; that is, true optimal solutions may rely on embedded loops. This is followed by a section recapping key differences and findings.

5.2 The original TRANSMax model

Models which attempt to address problems associated with mass transit have been explored since the advent of location modeling. Gleason (1975) offered a formulation for bus stop location which is based upon the Location Set Covering Problem developed by Toregas et al (1971). Church and ReVelle (1974) also proposed locating bus stops within the context of the Maximal Covering Location Problem. Current et. al. (1984, 1985) expanded transportation work by formulating both the Shortest Covering Path Problem and the Maximal Covering Shortest Path problems. Recent work has been advanced by researchers including that of Wu and Murray (2005), Matisziw et al. (2006), LaPorte et

al. (2005, 2011). Virtually all of the recent work is based upon the models first proposed by Current et. Al (1984,1985).

The TRANSMax model formulation takes a different tack; Curtin and Biba (2011) based their work on the Vajda (1961) framework for solving a TSP. They attempt to address the issue of transit based access by extending service access to include arcs as well as nodes and their model was explicitly developed for short access distances like those found in bus based transit systems. Their model is quite appealing from a transit planning perspective as it captures many of the elements of route coverage without explicitly specifying exact stop locations *a priori*. This gives transit planners flexibility in final stop location adjustments as well as being able to account for differences in service accessibility provided along long street segments as compared to shorter street segments.

This section gives the basic TRANSMax formulation. In the section following we propose a new TRANSMax formulation – TRANSMax II – in order to be more realistic with respect to transit planning. In particular this new version of TRANSMax is based upon a 'loop agnostic' approach so that real world routes can be created and use loops when such loops provide better levels of coverage. As was noted in Chapter 2 of this dissertation, the TRANSMax formulation can be adapted to meet several planning cases. Each of these cases can be easily captured through slight modifications of the modeling constraints. A detailed overview of these cases is given in Chapter 2; however, each of the cases in the original TRANSMax formulation are expressively prohibited from using loops as part of the solution due to the use of Vajda (1961) inspired constraints. The original TRANSMax notation and formulation are given below as:

i, j= index of nodes

r = index of sequence

o = the origin node for the route

t = the destination/terminus node for the route

R = the total number of segments, r that will be used

D = the maximum distance for the route

d_{ij} = the distance associated with the arc from node i to node j

A_{ij} = the service possible by utilizing an arc from node i to node j

M_i = the service value associated with node i

F_i = the set of nodes which can be reached by an arc from node i

T_j = the set of nodes which can be reached by an arc to node j

x_{ijr} = 1 if an arc from node i to node j on segment r is used, 0 otherwise

$$Maximize \quad Z = \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{r=1}^{R}(A_{ij} + M_i)x_{ijr} \tag{5.1}$$

Subject to

$$\sum_{i=1}^{m}\sum_{r=1}^{R} x_{ijr} \le 1 \;\forall\; j = 1,2,\ldots,m \tag{5.2}$$

$$\sum_{j=1}^{m}\sum_{r=1}^{R} x_{ijr} \le 1 \;\forall\; i = 1,2,\ldots,m \tag{5.3}$$

$$\sum_{i=1}^{m} x_{ijr} - \sum_{i=1}^{m} x_{ji(r+1)} = 0 \;\forall\; j = 1,2,\ldots,m; r = 1,2,\ldots,R-1 \tag{5.4}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m} x_{ijr} = 1 \;\forall\; r = 1,2,\ldots,R \tag{5.5}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{r=1}^{R} d_{ij}x_{ijr} \le D \tag{5.6}$$

$$x_{ijr} \in \{0,1\} \;\forall\; (i, j, r) \tag{5.7}$$

The complete discussion for the model can be found in Chapter 2 of this dissertation. For clarity here, a brief description of the model will be given in order to highlight key differences of the new TRANSMax II model that is subsequently formulated in this chapter. The objective (5.1) in the model maximizes arc service, A_{ij}, and node service,

M_i. Constraint (5.2) specifies that each node *j* can only be entered once across all sequences in *R* and constraint (5.3) specifies that each node *i* can only be departed once across all sequences in *R*. Taken together these constraints ensure that no sub-tours will form as they preclude the possibility of a tour from even forming. Constraint (5.4) is a balance constraint which ensures that if a node is entered on sequence *r* then on sequence $r+1$ that node must be departed. Constraint (5.5) specifies that for each sequence *r*, exactly one arc may be used. This ensures that there are not multiple arcs used for each sequence as this would not result in a connected path. Constraint (5.6) specifies that the total length of the covering path cannot exceed a maximum length, *D*. Constraints (5.7) are the binary restrictions on the decision variables x_{ijr}. The important aspect to note with respect to this formulation is that, as formulated, the original TRANSMax model cannot use loops/tours as part of a covering strategy. Thus, common route shapes that one finds in the 'real world' such as the figure eight, lollipop, and barbell (see Figure 5.1) can never form part of a solution determined by the original TRANSMax model. One of the key contributions of the revised formulation is to then allow for the possibility that loops may be used in such solutions. Therefore, the 'loop agnostic' – that is the form where loops can be used if it results in an improved objective – form of TRANSMax can be formulated as follows:

$$s_{ij} = \begin{cases} 1, \text{ if arc } (i, j) \text{ is served by the route} \\ 0, \text{ otherwise} \end{cases}$$

$$y_i = \begin{cases} 1, \text{ if node is served by the route} \\ 0, \text{ otherwise} \end{cases}$$

$$\text{Maximize } Z = \sum_i \sum_{j \in F_i, j>i} A_{ij} s_{ij} + \sum_{i \neq t} M_i y_i \qquad \textbf{(5.8)}$$

$$\sum_{j \in F_o} x_{oj1} = 1 \tag{5.9}$$

$$\sum_{j \in F_o} \sum_{r=1}^{R} x_{ojr} - \sum_{i \in T_o} \sum_{r=1}^{R} x_{ior} = 1 \tag{5.10}$$

$$\sum_{i \in T_t} \sum_{r=1}^{R} x_{itr} - \sum_{j \in F_t} \sum_{r=1}^{R} x_{tjr} = 1 \tag{5.11}$$

$$\sum_{i} \sum_{j \in F_i} x_{ijr} \le 1 \qquad \forall\ r = 1, 2, 3, \ldots, R \tag{5.12}$$

$$\sum_{j \in F_t} x_{tjr-1} - \sum_{j \in T_t} x_{jtr} \le 0 \qquad \forall\ r = 2, 3, 4, \ldots, R \tag{5.13}$$

$$\sum_{i \in T_j} x_{ijr} - \sum_{i \in F_j} x_{jir+1} = 0 \qquad \forall\ j,\ j \ne t,\ k = 1, 2, 3, \ldots, R-1 \tag{5.14}$$

$$\sum_{i} \sum_{j \in F_i} \sum_{r=1}^{R} d_{ij} x_{ijr} \le D \tag{5.15}$$

$$\sum_{r=1}^{R} \sum_{i \in T_j} x_{ijr} \ge y_i \quad \forall \quad j,\ j \ne t \tag{5.16}$$

$$\sum_{r=1}^{R} \sum_{i \in T_j} x_{ijr} + x_{jir} \ge s_{ij} \forall\ i,\ j,\ i < j \tag{5.17}$$

$$\begin{aligned} &x_{ijr} \in \{0,1\} \qquad \forall\ i,\ j \in F_i,\ r = 1, 2, 3, \ldots, R \\ &y_i \in \{0,1\} \qquad \forall\ i \\ &s_{ij} \in \{0,1\} \qquad \forall\ arc(i, j),\ i < j \end{aligned} \tag{5.18}$$

The objective function (5.8) for the revised TRANSMax model still maximizes arc and node service just as in the original TRANSMax formulation; however, we introduce two major changes in the formulation with respect to decision variables and constraints. The first major change is that two new variables are introduced to indicate whether an arc or a node is provided service. This is done through the variables s_{ij} and y_i which are used to represent arc service and node service respectively. If s_{ij} is utilized in the solution, this indicates service has been provided by arc(i , j). It should be explicitly noted here that s_{ij} represents service provided by the arc, not the direction of travel; the use of such variables is necessary to ensure that arc service is not double

counted as a loop could conceptually utilize the same arc. Thus, arc service is captured and counted exactly the same as in the original TRANSMax formulation. The use of y_i represents node service and will similarly ensure that node service is counted only once irrespective of how many times the node may be entered.

Constraint (5.9) requires that the path must leave the origin node. In this case, we specify this by stipulating that the arc must leave the origin and travel to a node *j* that is directly reachable from the origin node on the first sequence. Constraint (5.10) is a new type of constraint in the TRANSMax II model; in this case we allow a loop to form at the origin as long as the origin is departed exactly one more time than it is entered across all sequences. Constraint (5.11) is also a new type of constraint. Specifically, this constraint is used to ensure that the path terminates at the terminus node, *t*. The constraint is also unique in that it allows for a loop to form; in this case this is achieved by specifying that the terminus node must be entered exactly one more time than it is departed. Note that this constraint also relaxes the requirement in the original TRANSMax problem that exactly *R* sequences must be used. By allowing a loop to form and by ensuring it must be entered one more time than it is departed this ensures that a solution is not forced to conform to a preset number of sequences but rather can utilize as many sequences as required to ensure the best solution is obtained without needlessly extending path length.

Constraints of type (5.12) are also new in the formulation; in order to allow for the behavior above – that is, having a path where only the necessary sequences are used – we must specify that each segment must be less than or equal to one for each arc and sequence. If a sequence is necessary to form a connected path then it will be allowed to equal one. If it is not needed, however, the sequence is allowed to be zero and thus a

preset number of sequences does not force the solution to include unnecessary lengths. In conjunction with the binary requirements set for each variable, this constraint will then also ensure that no more than one arc will be selected for any given sequence. Constraints (5.13) are also new to the TRANSMax II formulation; these constraints are used to ensure that the last arc sequence used must enter the terminus node. Since constraint (5.11) only requires the terminus node to be visited it does not ensure that the path will return on the final segment. Since we have relaxed the requirement that a prespecified number of sequences must be used to that of utilizing only the number of sequences up to the maximum number of sequences, *R*, constraint (5.13) then ensures that the last sequence necessary to form an optimal path must be used to enter/return to the terminus node. Constraints of type (5.14) remain the same as in the original TRANSMax model and are used to ensure that, for each node that is not the origin or terminus node, if a node is entered on sequence *r* then that node must subsequently be left by sequence $r+1$. Constraint (5.15) also remains the same from the original TRANSMax model and specifies that the path cannot have a total length greater than *D*. Constraint (5.16) is a new constraint in TRANSMax II and defines coverage for nodes. In this case, a node is considered covered if it is departed by the path. Coverage is defined in this manner to match the original TRANSMax model; this can be readily seen in the objective function (5.1) where node service M_i is counted as a function of x_{ijr} – if x_{ijr} is utilized as part of the path in the original formulation then node service from *i* is counted.

Constraints of type (5.17) are used to define service provided by $arc(i, j)$. This is done through the use of the variable s_{ij}; note that this variable is designed such that the index value of *i* will always be less than the index value for *j* for this decision variable.

This is done so that the variable is able to capture the use of an arc but not double count travel which occurs in the opposite direction. This then ensures that arc service is not over-emphasized and is consistent with the original TRANSMax formulation, while still allowing an arc to be used in the formation of a loop if it is beneficial to do so. Constraints of type (5.18) are the binary restrictions on the decision variables.

This new TRANSMax II formulation is now 'loop agnostic' in the sense that loops are permitted to occur should they be utilized in an optimal solution. The formulation does not require loops to be used. It is important to note that the model also does not require the use of *R* sequences. In the original TRANSMax model this is a hard constraint and defining the number of arcs to be used a priori is not a particularly appealing approach in transit route design. The new TRANSMax II formulation allows for at most *R* arcs to be used while building in flexibility so that the route could potentially use less than the required number of arcs to find an optimal solution. This is particularly important as both the original TRANSMax and TRANSMax II formulations utilize a constraint which limits the maximum length of the path. By allowing flexibility in the number of sequences to be used it is possible that one may obtain a solution in which service is maximized using less than *R* sequences but which has an overall path length that is equal to or very near the maximum path length restriction. Although one could eliminate the path length constraint entirely, this may result in a more convoluted path that, while maximizing service, takes a very contorted route which in practicality is not terribly useful to a transit patron as they wish to get from one point to another as quickly as possible (Curtin and Biba, 2011). However, now that we have defined the new TRANSMax model, we can now obtain results and make direct comparisons with respect

to the performance of TRANSMax II compared to other models. This will be elaborated in the next section.

5.3 Computational Experience for the TRANSMax II Model

This section will highlight how the newly formulated TRANSMax II model performs in relation to other related models. We will analyze how the model performs with respect to the original TRANSMax formulation. Solution times as well as coverage results will be discussed. In particular, we wish to show that an optimal solution to the TRANSMax II model will mimic or exceed those of the original TRANSMax model given the same parameters. All results were generated in the same modeling environment detailed in other chapters: problems were formulated using the Mosel modeling language and solved using the Xpress solver of FICO Corporation. The remaining parts of this section are laid out as follows: first, a brief explanation will be given for the Richardson, TX study area as well as the metrics used to define arc service and node service will be discussed. Second, the solutions to several cases of the TRANSMax and TRANSMax II models as applied to this network will be given as well as a brief discussion of their performance with respect to time and the objective (service). The last section will offer concluding remarks with respect to performance of the model.

The choice to use Richardson, TX as a data set stems from the fact that Curtin and Biba (2011) utilize this as the data example for the TRANSMax model application. The network that we define is based upon the road network they provide which is centered on the Spring Valley DART (Dallas Area Rapid Transit) rail station. Figure 5.2 shows this network in detail; red representing the arcs allowed for possible routes and grey

representing the excluded surface streets. Owing to the fact that we do not have the data used by Curtin and Biba, we made an approximation of service values in the following manner.

The first step we took toward creating the Richardson, TX data set was to obtain the underlying geographic information. This was done through obtaining network data from the Open Street Map project. This data contains information such as road network, highway, bike trail, waterway, river, etc. In order to reduce the size of these dataset we removed all extraneous information that is unnecessary to the task of determining a covering path with respect to transit systems. As such, we removed information such as bike trails, walkways, interstates, off-ramps and on-ramps, rail-networks, etc. Once we had the underlying road network, we needed to quantify a rough estimate for household service. To accomplish this we crawled the Zillow.com website to obtain residential parcel data. We obtained information about the size of each home, how many bedrooms and bathrooms, as well as the estimated market price and parcel information. Given this data we then determined the centroids for these parcels. After determining the centroids, we determined the nearest surface street to the centroid as a proxy for the nearest walkable street location. A similar method was proposed by Biba, Curtin, and Manca (2010). The next step is to define the network for possible transit use. In this case we selected those arcs that matched the network given in Curtin and Biba (2011). This network is defined by Curtin and Biba to be composed of the major/minor arterial roads which would allow a bus to adequately make turns and pull off to the side of the street in order to make stops. Nodes were derived as a function of the intersections for each arc, and also matched those given in Curtin and Biba.

Once we attached the centroids of each parcel to their nearest street segment and defined our possible transit network, we calculated the distance from each attached centroid to their nearest node using ESRI's Network Analyst. This allowed us to calculate node service as a function of network distance; if one defined the maximum service distance provided by each node to be ¼ mile then you know exactly how many of these parcels are provided service by each node in the transit network that is defined. It should be noted that this differs from the approach taken by Curtin and Biba as they assigned node service using a random number – we wished to derive a quantifiable number based upon unique parcels. Arc service was calculated as a function of not only the possible number of stops per segment but also the population that could potentially be served along that segment. This is represented by (5.19) below:

$$A_{ij} = \frac{d_{ij}}{S} \times \frac{M_i + M_j}{2} \qquad \textbf{(5.19)}$$

where S represents the desired spacing between stops, d_{ij} represents the length of arc(i, j) and $\frac{M_i + M_j}{2}$ represents the average of the service provided by node i and node j. In this case we average the service provided by each node to stand as a proxy for the likely service that would be provided along an arc for any stop that is likely to be sited. The whole equation then represents the likely service that would be provided by an arc based upon the siting of potential stops along the arc. It should be noted that this is an expansion of the definition of service detailed by Curtin and Biba; they defined arc service as the number of potential stops whereas we attempt to account for service that would be a function of the number of potential stops as well as the likely number of households that could be serviced by those stops.

The next portion of this section will detail the solutions to several cases of the TRANSMax and TRANSMax II models as applied to the network we defined above. Solutions to the original TRANSMax will be presented first followed by solutions to TRANSMax II. These solutions can then be compared with respect to both sequence use and service (the objective). Each formulation is compared to the other with the same origin and terminus nodes and a stop service distance, *S*, of 1200 ft. Figures 5.3 through 5.12 show several solutions to the TRANSMax model applied to the Richardson, TX network. Figures 5.13 through 5.25 show the solutions to the TRANSMax II formulation as applied to the Richardson, TX network utilizing the same parameters. The solution times, objective, and other associated parameters for the TRANSMax model are shown in Table 5.1 while those for the TRANSMax II model are given in Table 5.2.

Table 5.1 – Parameters and Solution Values for the TRANSMax Model

Origin Node	Terminus Node	Max Sequences	Sequences Used	Path (Miles)	Node Service	Arc Service	Total Service	Time (Sec)
6	79	10	10	5.001	232	513.02	745.02	1.62
6	79	20	20	10.215	647	1507.87	2154.87	2.48
6	79	30	30	11.877	970	1763.03	2733.03	13.93
27	79	10	10	4.149	80	129.85	209.85	1.50
27	79	15	15	10.215	375	529.89	904.89	1.76
27	79	25	25	11.886	662	1575.25	2237.25	2.89
27	79	30	30	11.916	786	1517.56	2303.56	17.57

Origin Node	Terminus Node	Max Sequences	Sequences Used	Path (Miles)	Node Service	Arc Service	Total Service	Time (Sec)
70	79	10	10	5.202	323	768.69	1091.69	1.48
70	79	15	15	8.487	462	1166.61	1628.61	1.70
70	79	20	20	10.709	611	1478.11	2089.11	2.71
70	79	30	30	11.896	904	1771.42	2675.42	36.35
27	21	10	10	4.450	143	358.31	501.31	1.45
27	21	20	20	9.497	489	1095.30	1584.30	1.80
27	21	30	30	11.994	774	1411.29	2185.29	9.50

Computational experience between both models proved to be statistically similar for most parameters. However, the TRANSMax II model did take significantly longer to solve for a few cases. The reason for this is the fact that, in terms of the formulation, the constraints are not as

Table 5.2 – Parameters and Solution Values for the TRANSMax II Model

Origin Node	Terminus Node	Max Sequences	Sequences Used	Path (Miles)	Node Service	Arc Service	Total Service	Time (Sec)
6	79	10	10	5.001	232	513.02	745.02	0.85
6	79	20	20	10.215	647	1507.87	2154.87	2.86
6	79	30	30	11.994	926	1808.47	2734.47	274.35
27	79	10	10	4.149	80	129.85	209.85	0.77
27	79	15	15	7.113	375	529.89	904.89	1.23
27	79	25	25	11.886	662	1575.25	2237.25	1.81
27	79	30	30	11.991	786	1543.20	2329.20	111.40

Origin Node	Terminus Node	Max Sequences	Sequences Used	Path (Miles)	Node Service	Arc Service	Total Service	Time (Sec)
70	79	10	10	5.202	323	768.69	1091.69	0.75
70	79	15	15	8.232	424	1217.76	1641.76	1.03
70	79	20	20	10.774	611	1608.82	2219.82	2.18
70	79	30	29	11.954	897	1810.64	2707.64	2544.58
27	21	10	9	4.517	143	358.85	501.85	0.81
27	21	20	20	9.497	489	1095.30	1584.30	1.30
27	21	30	29	11.887	760	1462.13	2222.13	81.00

tight. This is due to the fact that the TRANSMax II model allows for the formation of loops/tours as well as the fact that it does not require that the maximum number of sequences are actually used. It can be observed from results shown in Table 5.2 that as the route approaches the maximum path length (12 miles) there is a tendency to include a loop or tour in the optimal solution. The case where the number of sequences used is less than the maximum number of sequences also highlights the fact that there may be an area which has a higher potential service but is only possible to be visited by a certain number of arcs, or the path distance limit makes the problem behave somewhat like a knapsack problem (i.e. there is a limited space and one must find the optimal number of items to include based upon their size, value, etc.). The results do show that the TRANSMax II problem uses the flexibility we have built into the problem to achieve solutions that are equal to or superior to those derived from the original TRANSMax formulation. This is particularly important as this shows that a loop agnostic strategy is imperative with respect to finding truly optimal solutions to covering path type problems, and verifies the work of Niblett and Church (2016) which highlighted the use of loops as an optimal covering strategy with respect to the Maximal Covering Shortest Path problem.

5.4 Conclusions and Future Work

Up to this point, this chapter has presented several aspects and issues that have arisen with the TRANSMax model formulation. We have seen that the TRANSMax model formulation takes a unique approach with respect to traditional covering path problem formulations based upon the MCSP. We have also defined the TRANSMax model; we have shown how the formulation prevents the use of loops or tours, as well as how the model requires the use of *R* sequences. A new model formulation was proposed, TRANSMax II, which addresses this deficiency as well as relaxes the requirement that exactly *R* sequences be used. This new model utilizes a generalized form of the Vajda construct. This new formulation was then compared to solutions determined by the original TRANSMax formulation. We also defined a new metric for arc service as well as explained how our test network was derived and data acquired. This section presents a brief recap of these findings as well as possible research directions for the TRANSMax problem.

This chapter notes that traditional MCSP based problems are defined such that coverage is defined to be within some service standard and maximized and overall path length is minimized. These two competing objectives are often proportionally weighted so that emphasis placed on coverage is proportional to emphasis placed on path length. Typically these models only treat nodes as demands which must be covered while arcs are merely used to define a possible network on which the path must be defined. The TRANSMax model is unique in that not only is service defined as a function of both nodes and arcs, the formulation is formulated using a Vajda (1961) based sub-tour prevention framework. Unfortunately, this framework results in sub-optimal solutions as

it is possible a loop or tour could form part of an optimal solution. In order to overcome this issue as well as allow for more robust solutions to be used the TRANSMax II model was formulated.

The TRANSMax II model formulation not only permits the use of loops as a covering strategy by being 'loop agnostic,' it also relaxes the requirement that *R* arcs be used. Instead the formulation specifies a maximum number of arcs that can be used while allowing the model to use less should a better solution exist. This was proven through the utilization of the Richardson, TX based network. This network was defined based upon the original network given in Curtin and Biba (2011). Information regarding household parcels was obtained through Zillow.com and a novel method for attaching these households to the transit network was adopted similar to what was described in Biba et. al. (2010). After this dataset was created, we defined a new approach with respect to calculating arc service based upon not only the number of potential stops but also the likely service that each of these potential stops could provide. This allows for a more realistic problem in that service is directly tied to a stop location and not merely the number of stops; this then helps to avoid cases where a long country road may have the same level of service provided by a road next to a park for example.

Following the discussion of how we defined our network and service data, we presented results to both the original TRANSMax and the new TRANSMax II models. Several origin and terminus nodes were used as well as a wide variation in maximum sequence values. The results to these problems show that the TRANSMax II formulation finds solutions which are equal to or better than those obtained for the original TRANSMax model. The results also indicate that, in general, the TRANSMax II

formulation is just as efficient as the original TRANSMax with respect to solution times. We also noted that cases in which the TRANSMax II formulation took longer to solve often corresponded to cases in which the maximum path distance constraint was binding or in cases where fewer than the maximum number of sequences could be used which resulted in a superior solution. This verifies the work of Niblett and Church (2016) which shows that loops will be used in many optimal solutions (when allowed) as well as the fact that allowing flexibility in the formulation with respect to relaxing the requirement that exactly *R* sequences be used can determine solutions with a significantly higher level of service.

Future work with respect to the TRANSMax II formulation includes incorporating a weighted distance objective into the formulation. Preliminary results for this kind of formulation indicate that the TRANSMax II model can replicate results derived for the MCSP. Other significant attributes that can be explored include defining an arc service value for MCSP based models with respect to transit routing as this may offer further insight into these kinds of problems. One motivating factor to approach the solutions to MCSP based models lies in the fact that, while the TRANSMax II model solves reasonably quickly in this example, applying it to larger networks results in a significant increase in variables and constraints. Orman and Williams (2006) also indicate this would be a fruitful approach, particularly as their work explored computational experiences based upon differing TSP formulations, a problem that is related to both the MCSP and TRANSMax.

Therefore, we have defined, formulated, and solved a new form for the TRANSMax II problem and observed that loops appear in optimal solutions in certain

cases. The next chapter of this dissertation will focus on moving from a "loop agnostic" strategy to a "loop encouraging" strategy as well as defining how one can determine optimal covering paths in a bi-directional manner rather than assuming a MCSP can simply be implemented in the reverse order.

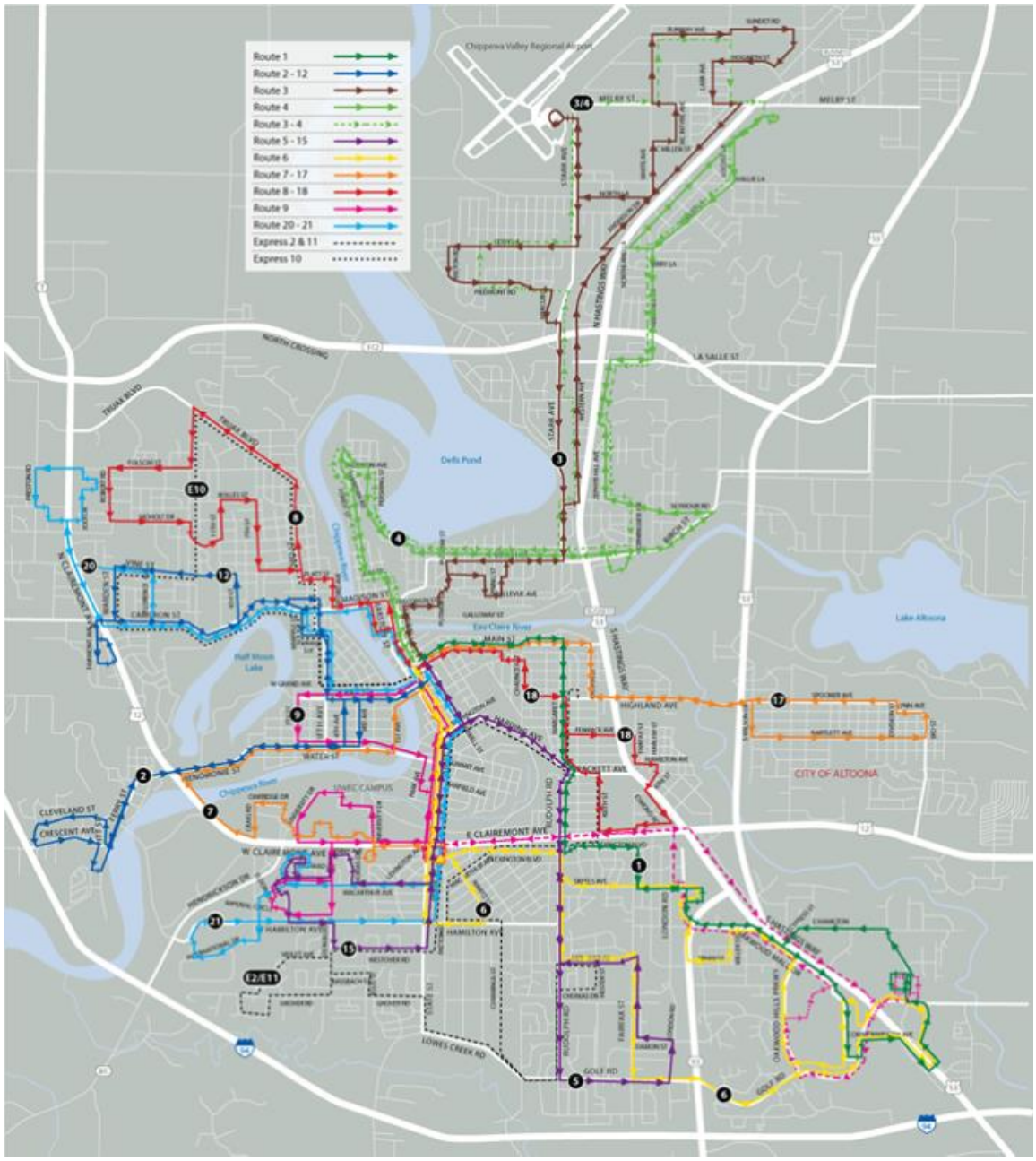


Figure 5.1 - Route Map Highlighting the use of loops in Eau Claire, WI. Permission to use this image has graciously been given by the City of Eau Claire.

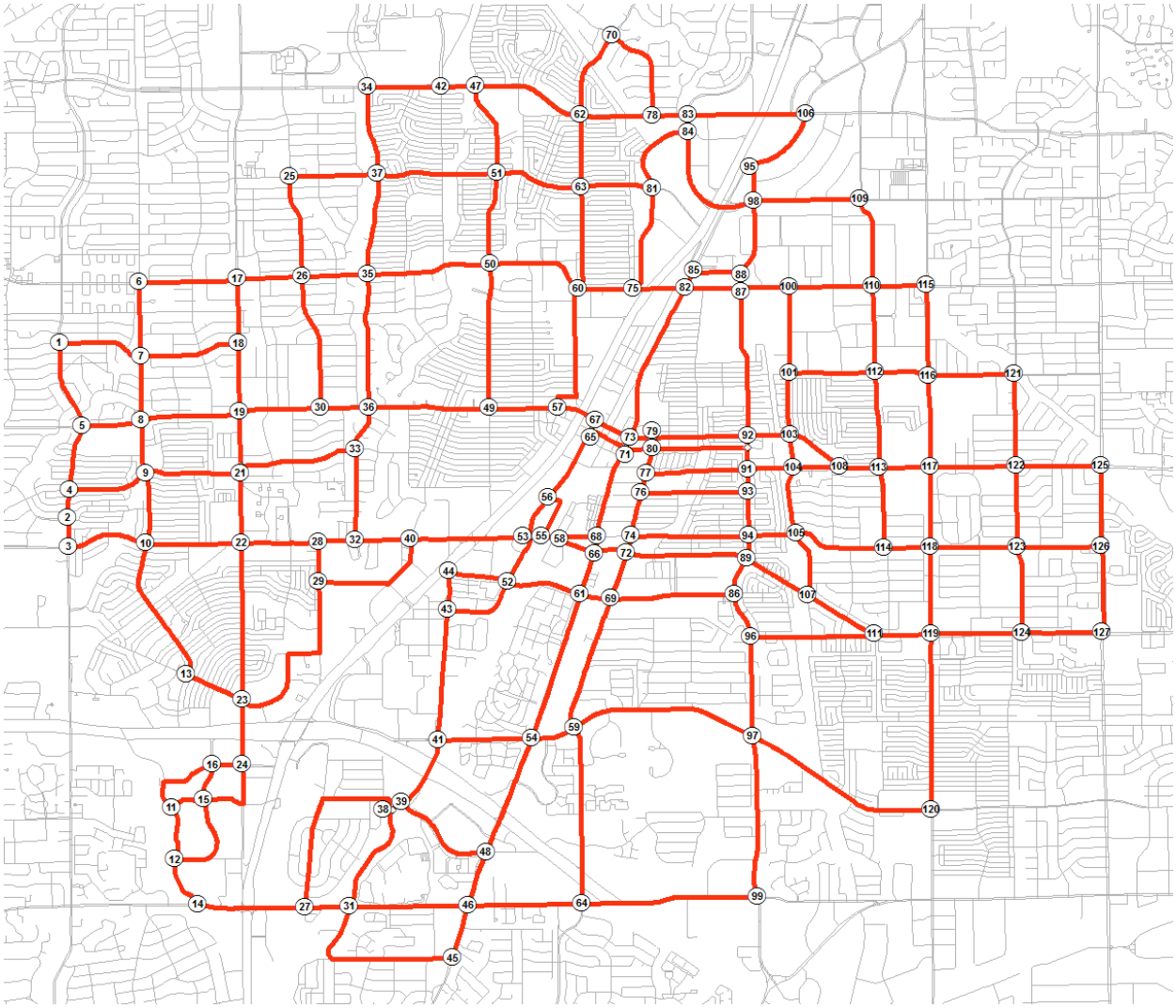


Figure 5.2 - The Richardson, TX transit network (red) centered around the DART Spring Street Station (Node 55); the grey lines are the surface streets that were not considered as part of the possible transit network.

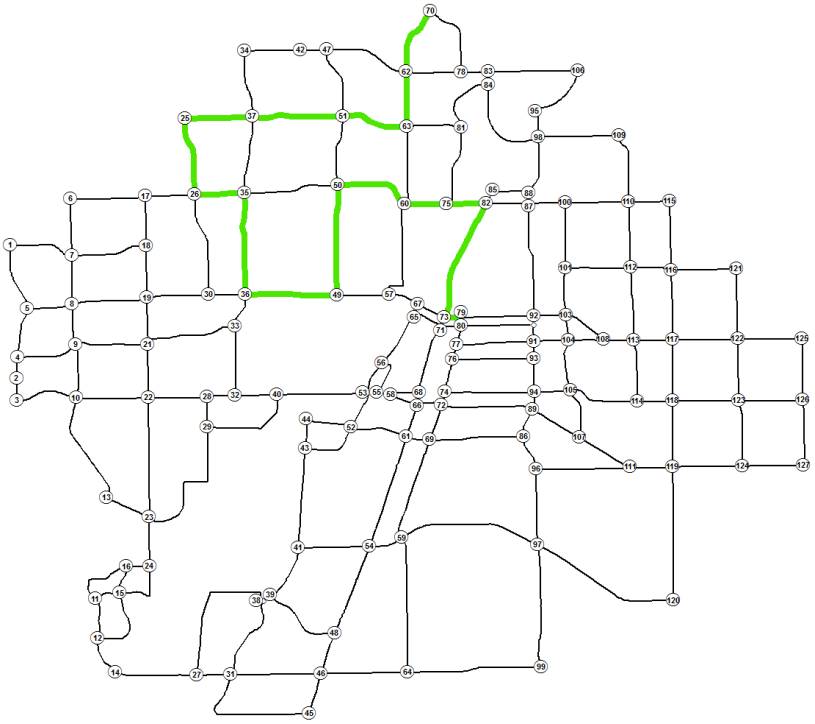


Figure 5.3 - Solution to the TRANSMax model with an origin at node 70, a destination at node 79, the number of sequences set to 15, and a maximum path length of 12 miles

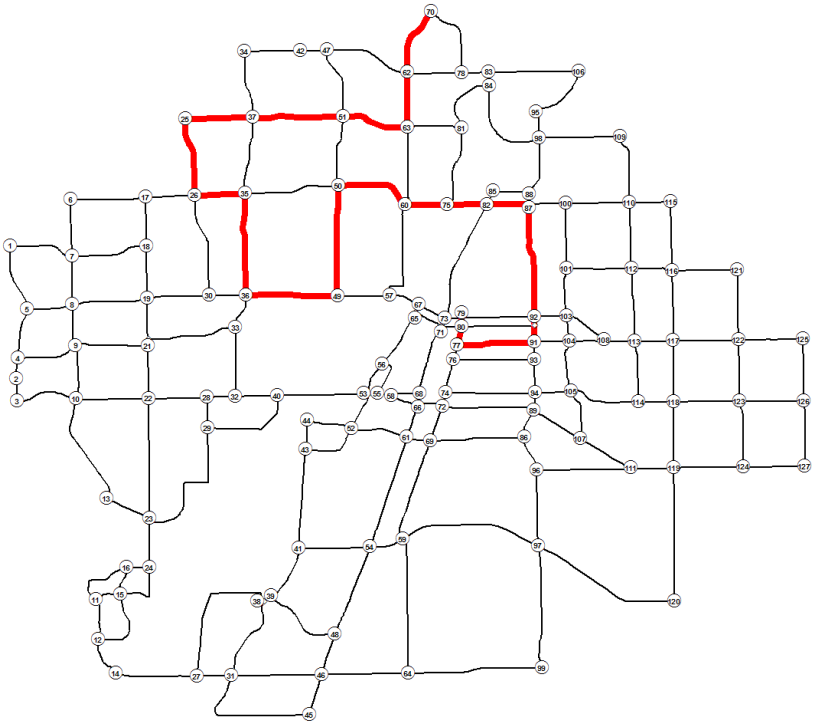


Figure 5.4 - Solution to the TRANSMax model with an origin at node 70, a destination at node 79, the number of sequences set to 20, and a maximum path length of 12 miles

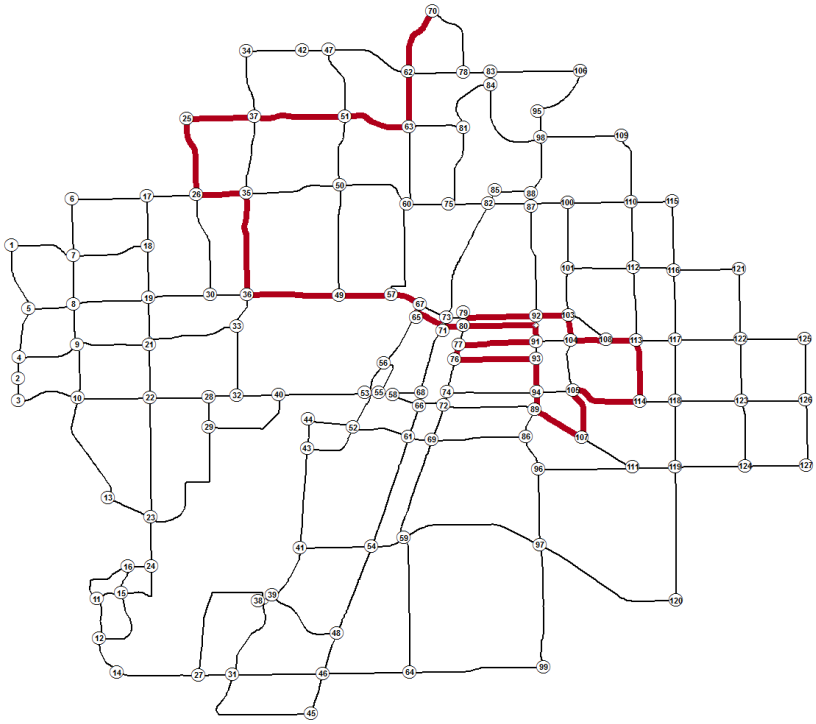


Figure 5.5 - Solution to the TRANSMax model with an origin at node 70, a destination at node 79, the number of sequences set to 30, and a maximum path length of 12 miles

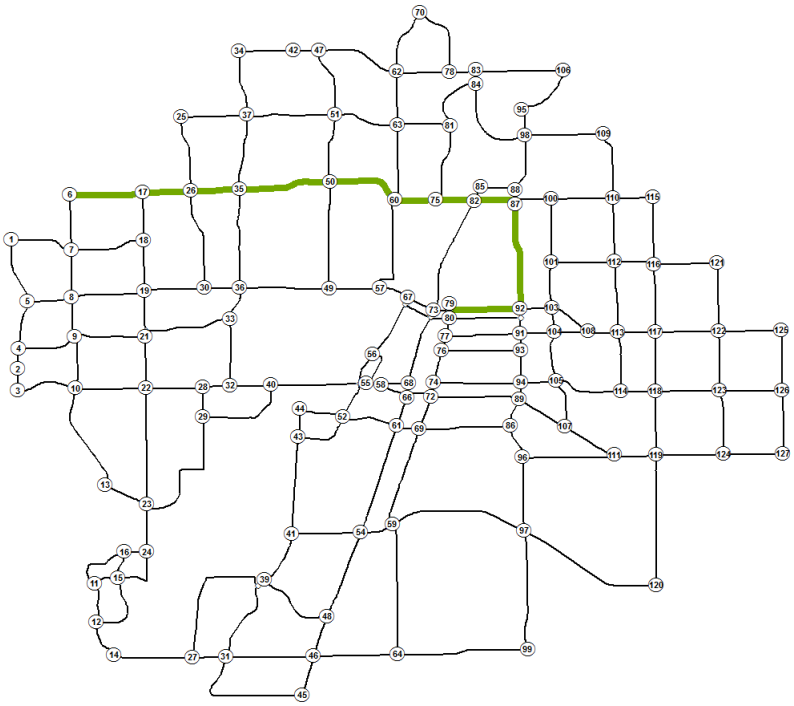


Figure 5.6 - Solution to the TRANSMax model with an origin at node 6, a destination at node 79, the number of sequences set to 10, and a maximum path length of 12 miles

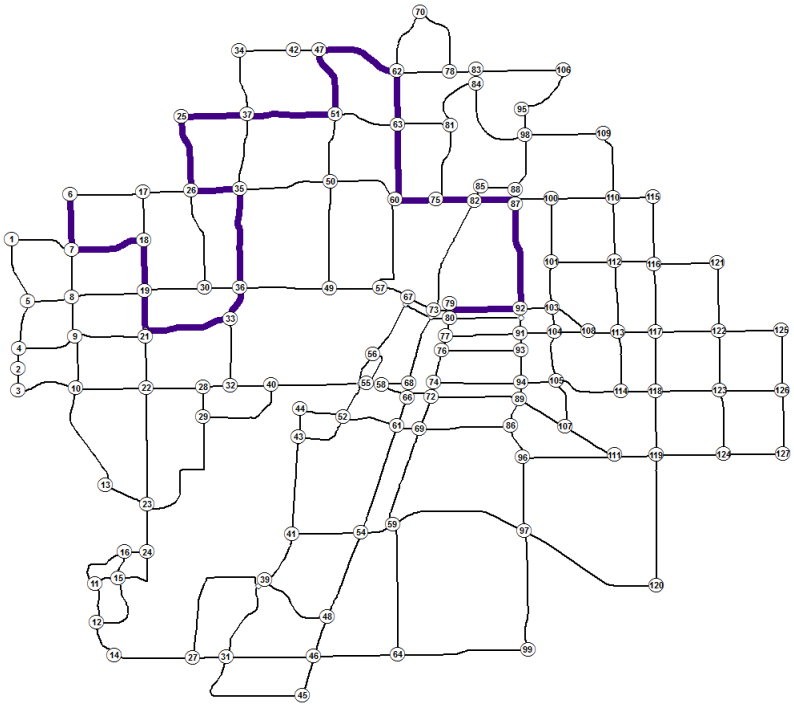


Figure 5.7 - Solution to the TRANSMax model with an origin at node 6, a destination at node 79, the number of sequences set to 20, and a maximum path length of 12 miles

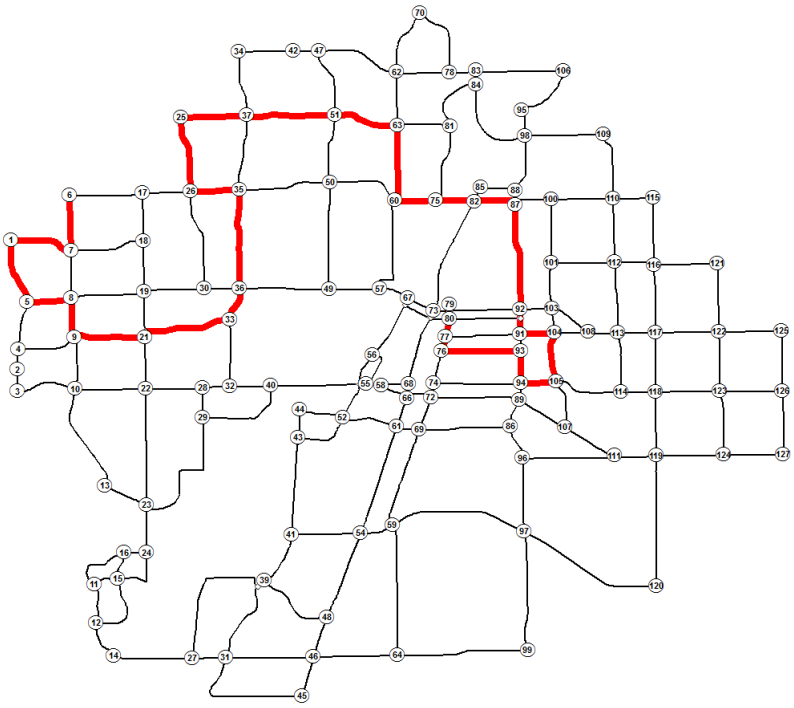


Figure 5.8 - Solution to the TRANSMax model with an origin at node 6, a destination at node 79, the number of sequences set to 30, and a maximum path length of 12 miles

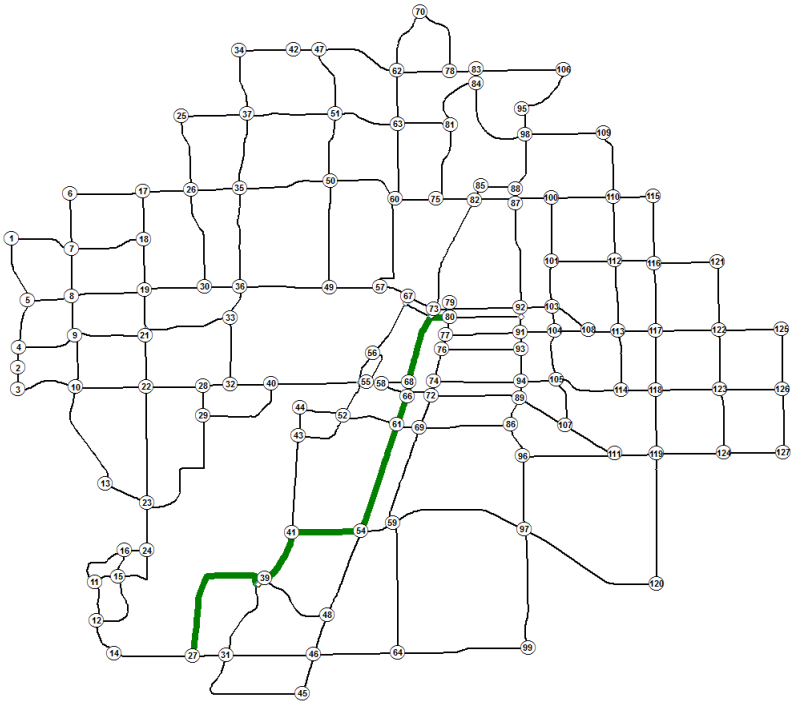


Figure 5.9 - Solution to the TRANSMax model with an origin at node 27, a destination at node 79, the number of sequences set to 10, and a maximum path length of 12 miles

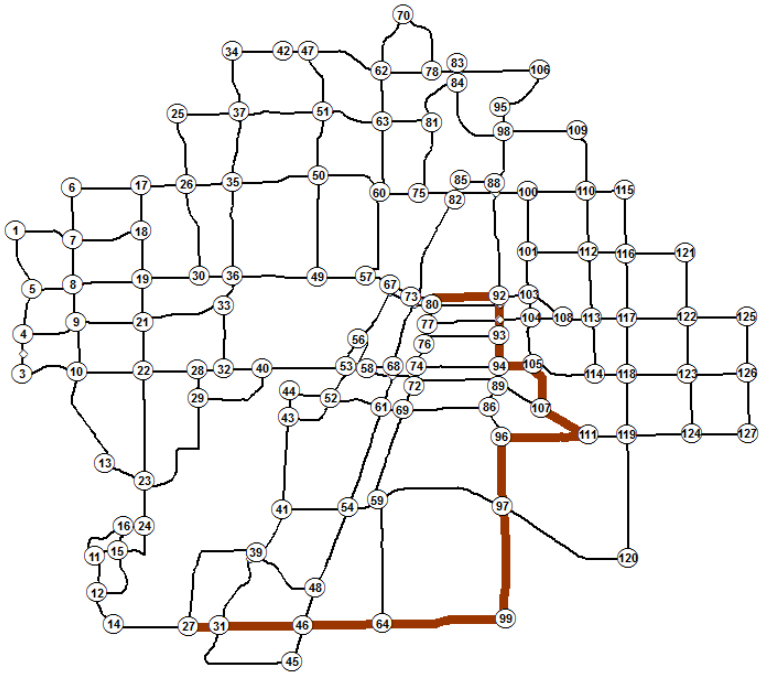


Figure 5.10 - Solution to the TRANSMax model with an origin at node 27, a destination at node 79, the number of sequences set to 15, and a maximum path length of 12 miles

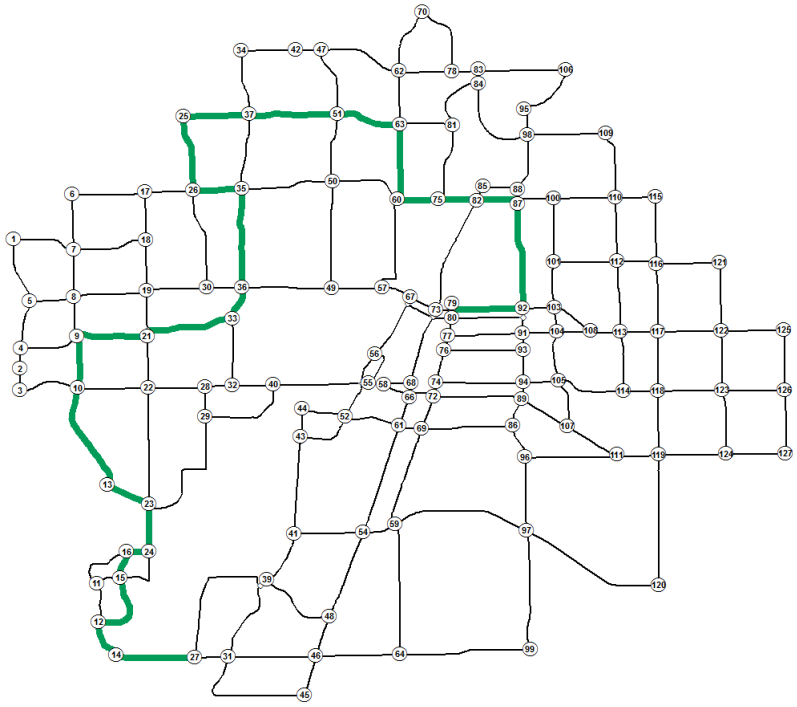


Figure 5.11 - Solution to the TRANSMax model with an origin at node 27, a destination at node 79, the number of sequences set to 25, and a maximum path length of 12 miles

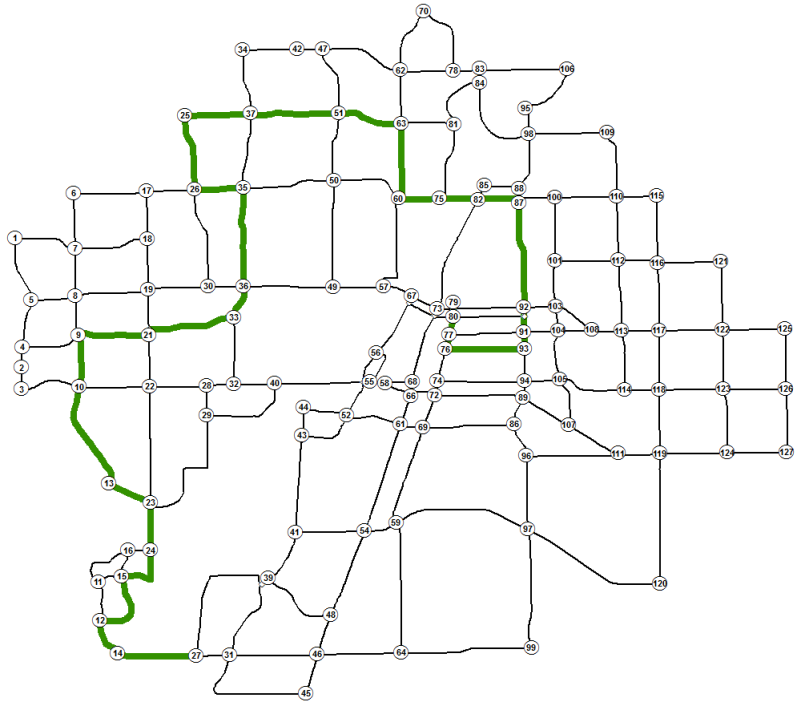


Figure 5.12 - Solution to the TRANSMax model with an origin at node 27, a destination at node 79, the number of sequences set to 30, and a maximum path length of 12 miles

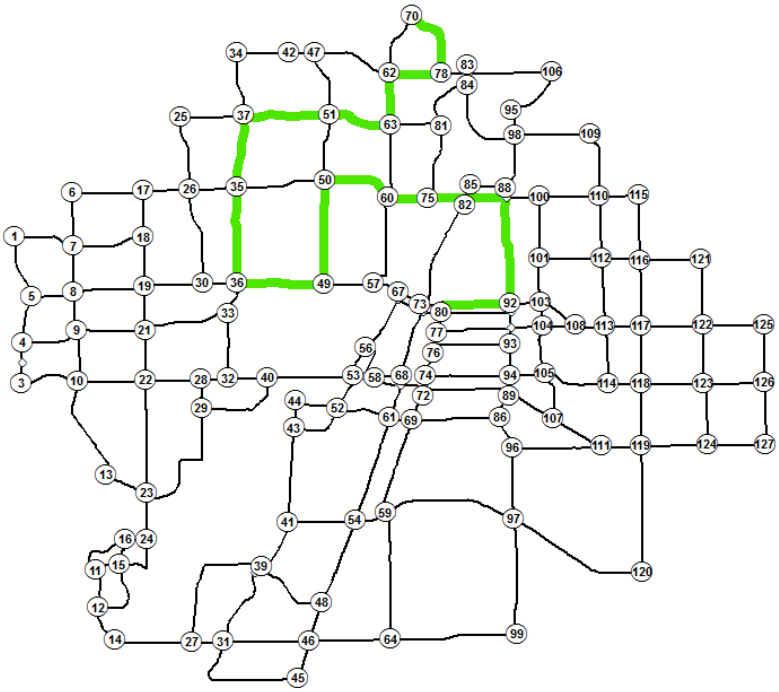


Figure 5.13 - Solution to the TRANSMax model with an origin at node 70, a destination at node 79, the number of sequences set to 15, and a maximum path length of 12 miles

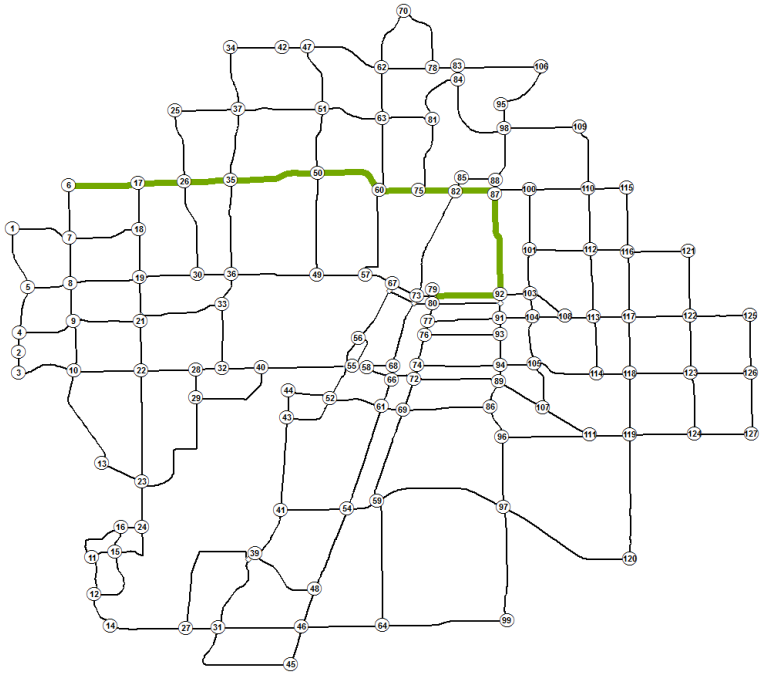


Figure 5.14 - Solution to the TRANSMax II model with an origin at node 6, a destination at node 79, the maximum number of sequences set to 10, and a maximum path length of 12 miles

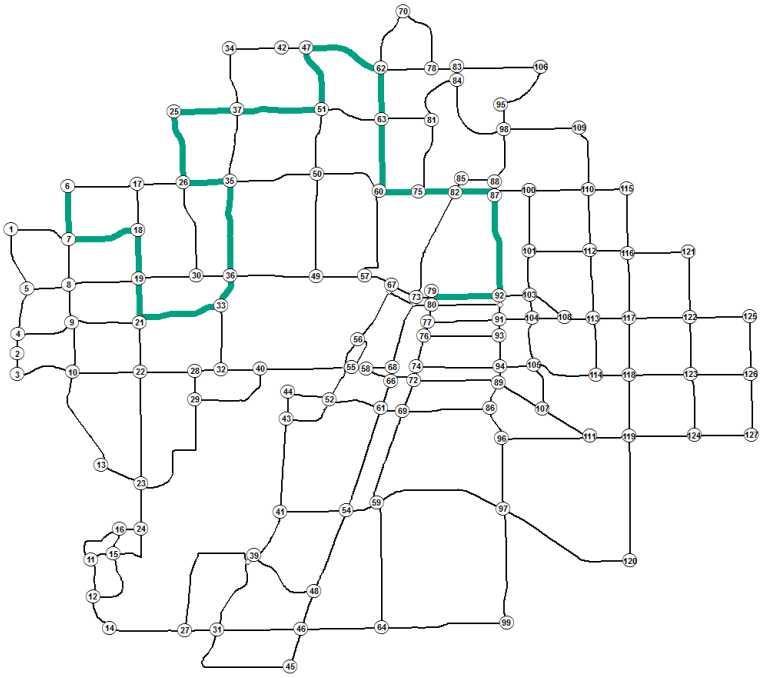


Figure 5.15 - Solution to the TRANSMax II model with an origin at node 6, a destination at node 79, the maximum number of sequences set to 20, and a maximum path length of 12 miles

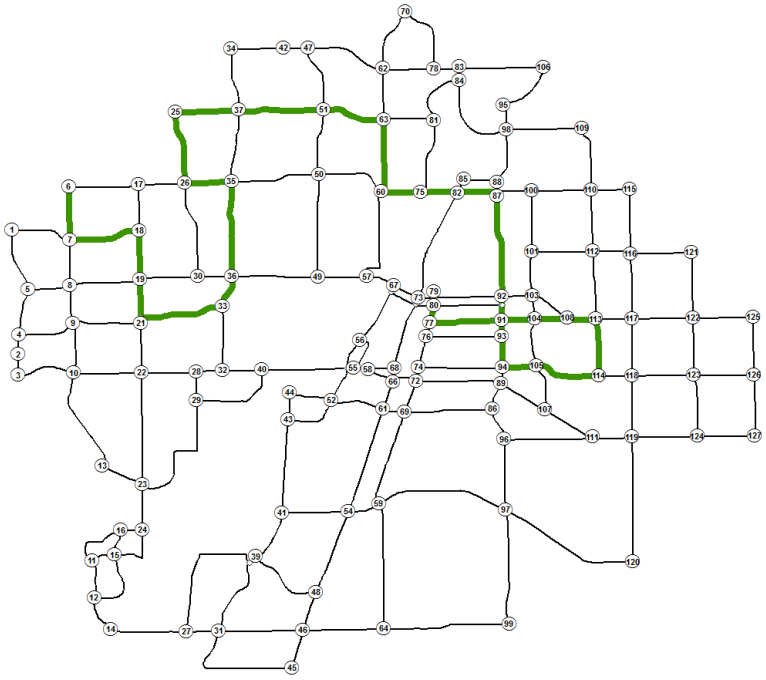


Figure 5.16 - Solution to the TRANSMax II model with an origin at node 6, a destination at node 79, the maximum number of sequences set to 30, and a maximum path length of 12 miles

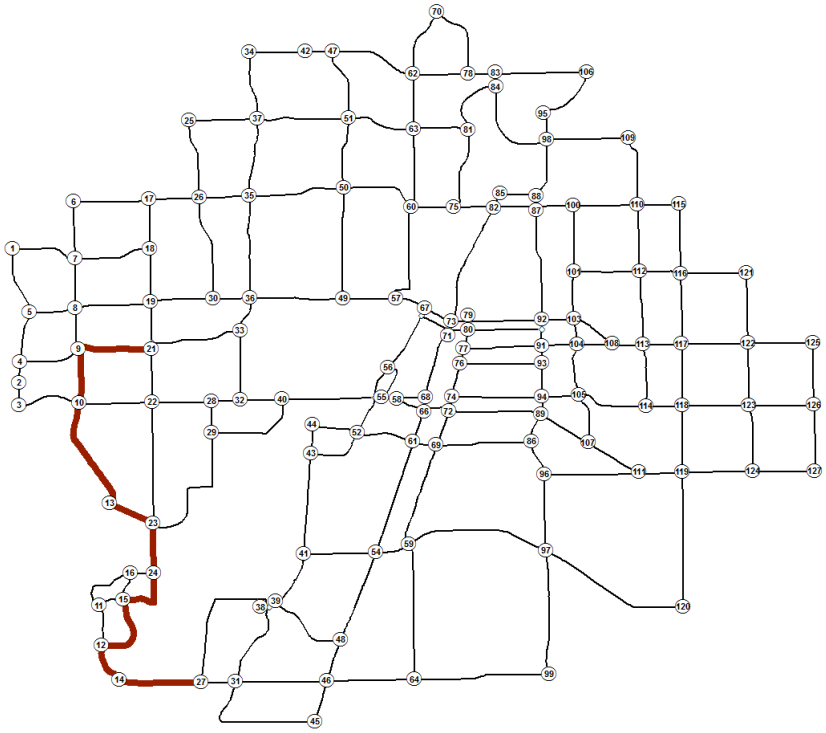


Figure 5.17 - Solution to the TRANSMax II model with an origin at node 27, a destination at node 21, the maximum number of sequences set to 10, and a maximum path length of 12 miles

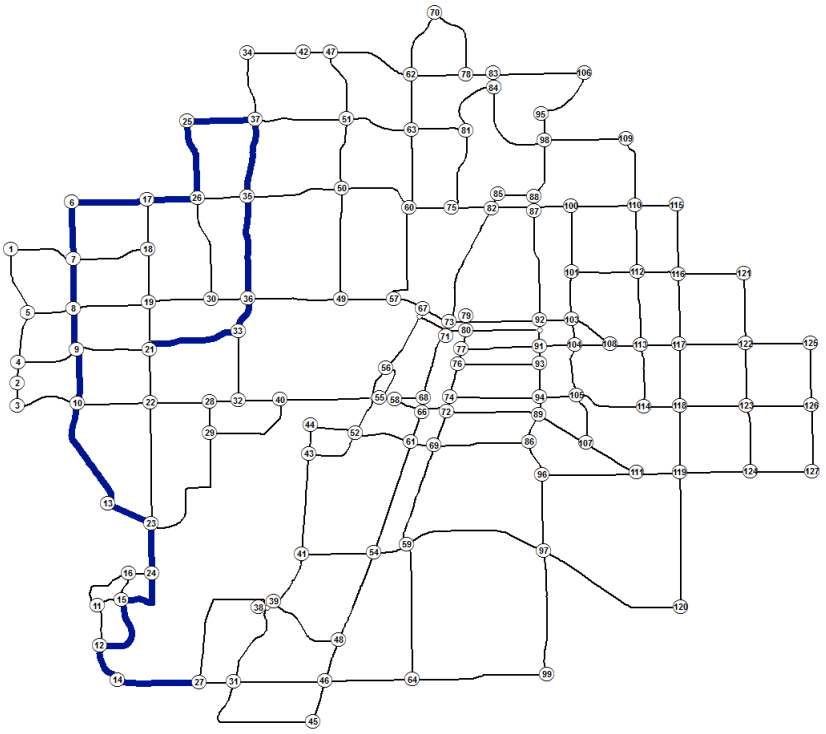


Figure 5.18 - Solution to the TRANSMax II model with an origin at node 27, a destination at node 21, the maximum number of sequences set to 20, and a maximum path length of 12 miles

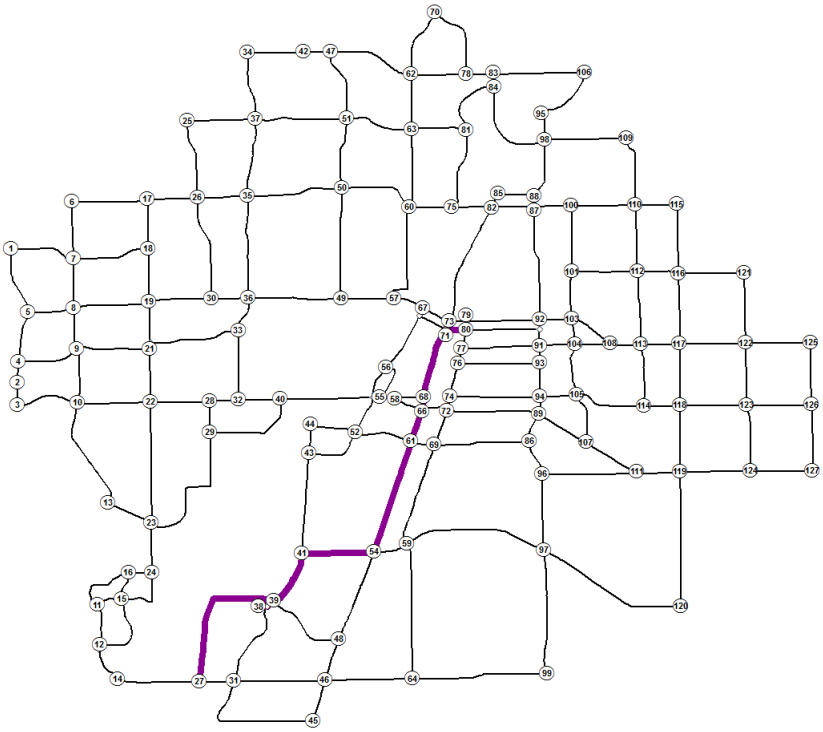


Figure 5.19 - Solution to the TRANSMax II model with an origin at node 27, a destination at node 79, the maximum number of sequences set to 10, and a maximum path length of 12 miles

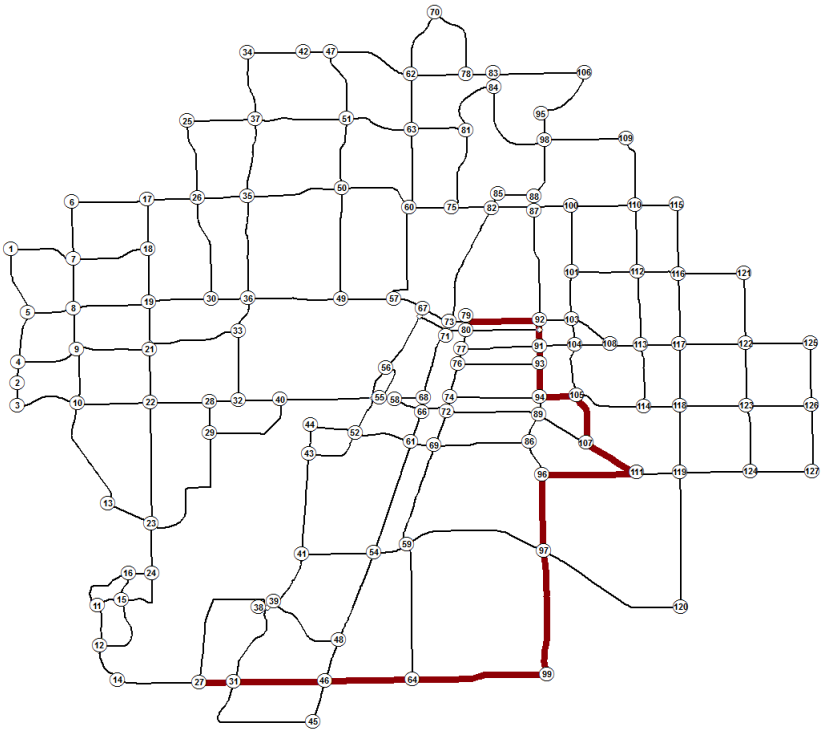


Figure 5.20 - Solution to the TRANSMax II model with an origin at node 27, a destination at node 79, the maximum number of sequences set to 15, and a maximum path length of 12 miles

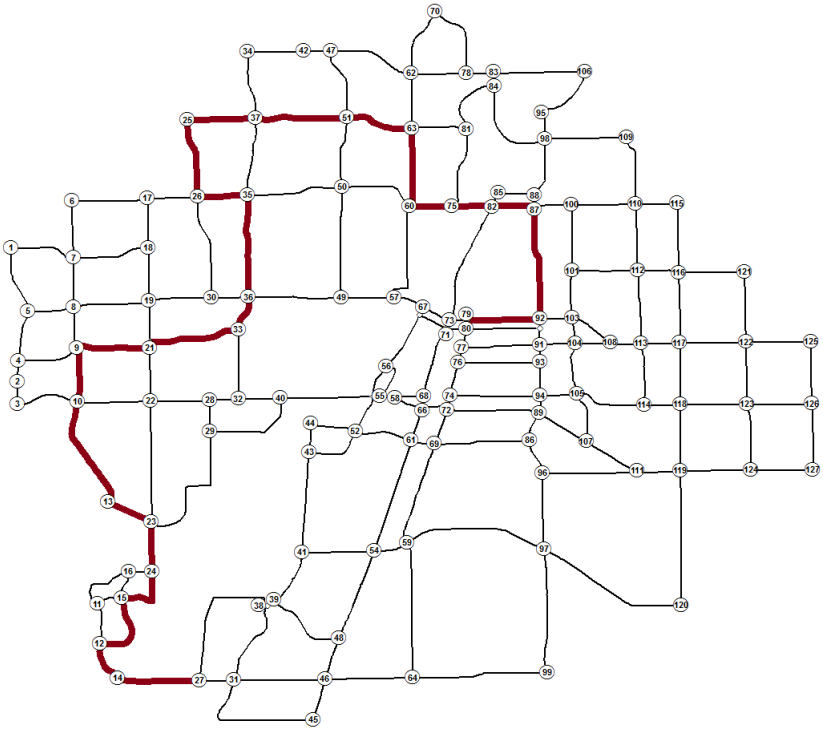


Figure 5.21 - Solution to the TRANSMax II model with an origin at node 27, a destination at node 79, the maximum number of sequences set to 25, and a maximum path length of 12 miles

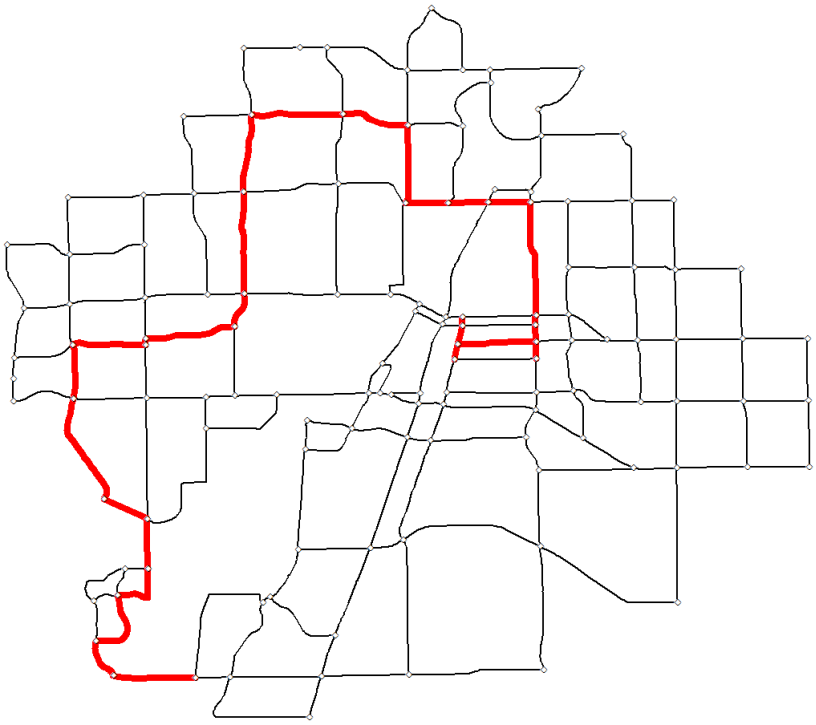


Figure 5.22 - Solution to the TRANSMax II model with an origin at node 27, a destination at node 79, the maximum number of sequences set to 30, and a maximum path length of 12 miles. Labels have been omitted to highlight the use of loops

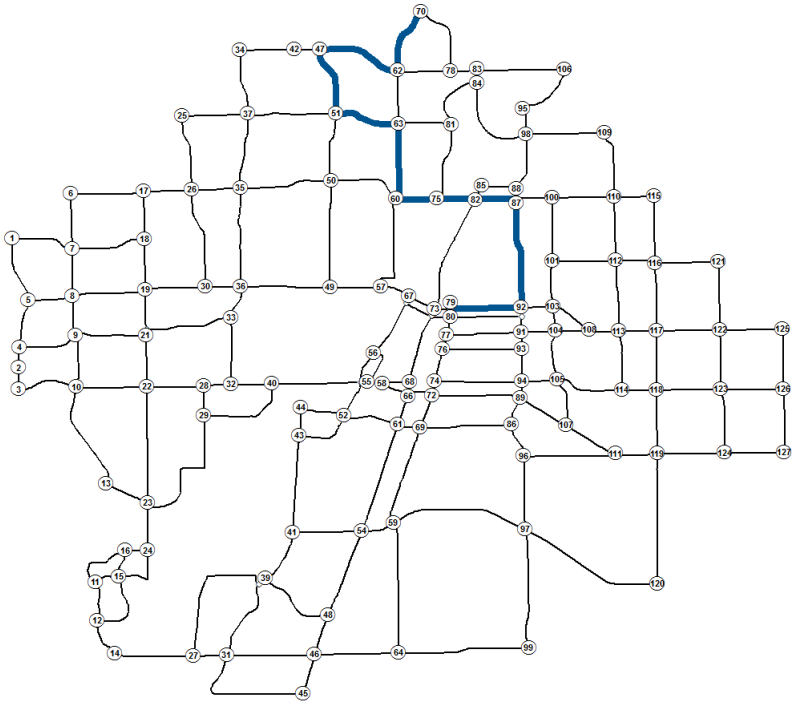


Figure 5.23 - Solution to the TRANSMax II model with an origin at node 70, a destination at node 79, the maximum number of sequences set to 10, and a maximum path length of 12 miles

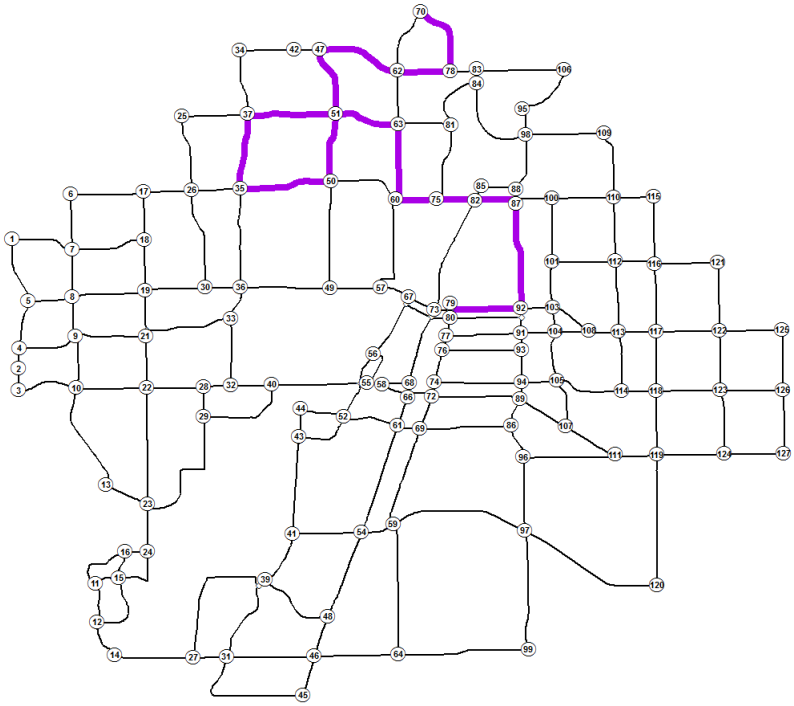


Figure 5.24 - Solution to the TRANSMax II model with an origin at node 70, a destination at node 79, the maximum number of sequences set to 15, and a maximum path length of 12 miles

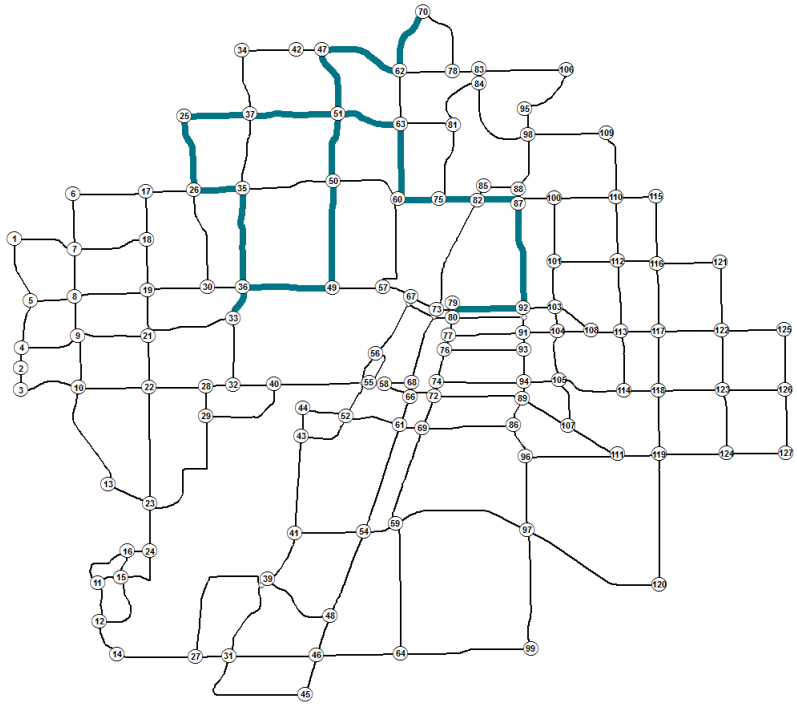


Figure 5.25 - Solution to the TRANSMax II model with an origin at node 70, a destination at node 79, the maximum number of sequences set to 20, and a maximum path length of 12 miles

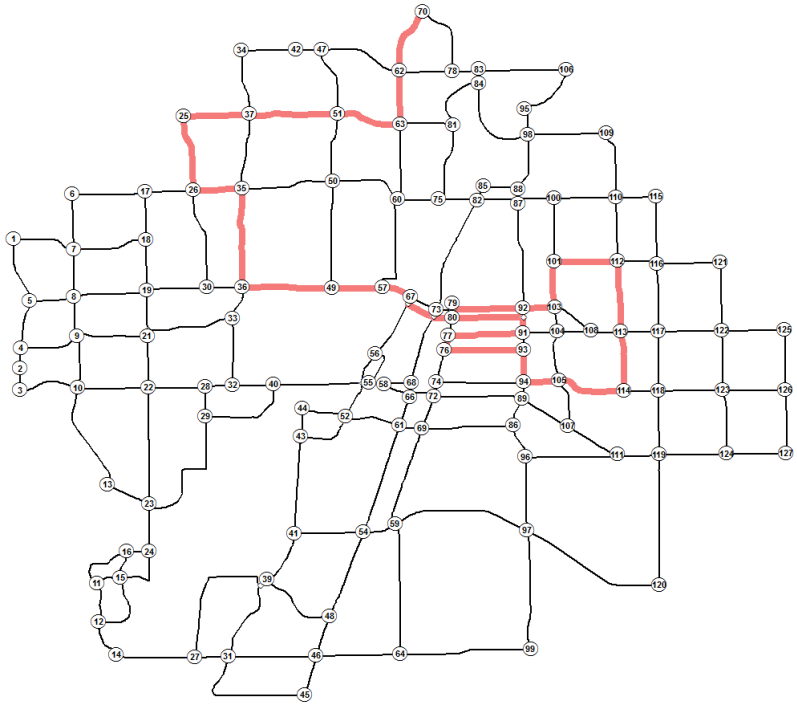


Figure 5.26 - Solution to the TRANSMax II model with an origin at node 70, a destination at node 79, the maximum number of sequences set to 30, and a maximum path length of 12 miles

Chapter 6

6.1 Introduction

This chapter represents a major departure from the previous chapters of this dissertation, particularly chapters 3, 4, and 5. Chapter 3 presented a new model, NR-MCSP, for solving the maximal covering shortest path problem. This new model was termed "loop agnostic." That is, this model allowed loops to be used in path formation whenever embedded loops help produce a better solution. That chapter also described why the original streamlined maximum population shortest path (MPSP) model of Current et al. (1985) could not be extended to a "loop agnostic" form, although the new NR-MCSP model proposed in that chapter can be used for solving a loop agnostic form of the MPSP problem. Chapter 4 presented a "substitution/swapping" based heuristic for solving the new NR-MCSP model. This new heuristic borrows heavily upon past work involving substitution/insertion heuristics that was initially begun by Lin (1965) for the Travelling Salesman Problem. What is entirely new about this heuristic is that it is capable of substituting route components which can represent the building blocks of loops, even out and back loops, into a shortest path or route. Results from both Chapters 3 and 4 demonstrated that embedded loops can be present in both optimal and heuristically derived local optimal solutions.

Chapter 5 was based upon taking a different tack in structuring and solving the original MCSP. Recall that Curtin and Biba developed a new formulation of the MCSP that was based upon the TSP model of Vajda (1961). Their new model was called TRANSMax. By using the underlying form of Vajda, their new model prevents sub-tours

from appearing in a solution even when explicit sub-tour elimination constraints are not used (as opposed to constraints based upon Dantzig, Fulkerson, & Johnson). Chapter 5 presented a new model form that represents an extension of TRANSMax which allows loops to be formed within a solution as long as such loops are attached to the path. This new structure demonstrates that there can be other alternate pathways in which the MCSP model can be formulated for the general case (i.e. allow embedded loops) while utilizing a Vajda inspired framework.

This chapter explores a new, more general form of the MCSP problem. The basic assumption in applying the classic MCSP for transit and other route design problems is that the path will be traveled in both directions when needed. That is, a bus, using a MCSP path, will traverse all of the arcs along the path from the origin to the destination, and then reverse its direction and travel those same arcs beginning at the destination and heading towards the origin. Sometimes such a complete reversal is not possible due to the use of one-way streets. But, when that occurs then the return pathway simply uses the neighboring one-way streets heading in the appropriate direction, so that the return, inbound trip veers no more than a block or so away from the outbound trip or path. This also means that in the more flexible NR-MCSP problem, it is assumed that a loop in a path will be traversed in both directions (once outbound and once inbound). This chapter is based upon the assumption that the outbound and inbound path need not be entirely the same. That is, although certain route segments of a covering path may be the same for both the outbound and inbound path, some segments may be different. We can define this new form of the Maximal Covering Shortest Path problem as follows:

Find an "outbound" path starting at the origin and ending at the destination and an "inbound" path starting at the destination and ending at the origin, that in combination maximizes coverage, minimizes total distance traveled (in both directions) and where at least some portion, Ω *, of the outbound and inbound routes use the same street or arc segments.*

This problem will be defined as the bi-directional maximal covering shortest path (BD-MCSP) problem. This problem is somewhat related to the multi-path maximal covering shortest path problem of Boffey and Narula (1998). In their model, they assume that such multiple routes (or paths) start at the same origin and end at the same destination, and each route or pathway is independent of the other, except that they may be constrained to be in the same region or corridor stretching between the origin and destination. They make no assumption about direction of travel, just as in Current et al (1984, 1985). Presumably, each pathway will be traveled in both inbound and outbound directions when they are applied. Total coverage equals the total sum of coverage provided by all of the paths. The bi-directional MCSP problem is similar to that of Boffey and Narula in that it involves determining two directed paths (one inbound and one outbound) but differs in that the inbound and outbound paths are encouraged to coincide whenever possible, whereas there was no such attempt in Boffey and Narula. The basic idea is that service coverage may be extended by expanding one direction or other direction of travel, without seriously increasing route distance. This appears to be a strategy used by many small to medium sized cities in transit route design, a point that will be described in more

detail in the next section. We can also define a variant of the BD-MCSP problem as follows:

> *Find an "outbound" path starting at the origin and ending at the destination and an "inbound" path starting at the destination and ending at the origin, that in combination maximizes coverage, minimizes total distance traveled (in both directions) and where it is encouraged that a portion of the outbound and inbound routes visit the same nodes or intersections.*

Whereas the first form ensures a certain portion of the street segments coincide for the outbound and inbound paths, this second form attempts to encourage both outbound and inbound paths to coincide at intersections, when possible.

The remainder of this chapter is organized as follows. The next section will present several examples of transit system designs, which demonstrate that transit planners use bi-directional paths to provide high levels of access coverage while at the same time keeping path/route mileage as low as possible. Particular emphasis will be placed on defining how current models are not capable of generating such route alternatives and why the models proposed in this chapter represent a step forward in not only transit route modeling problems but also the broader class of covering path problems. The following section will define two mathematical formulations which represent the two design problems defined above. This section will focus on explaining the notation for the model as well as defining the objectives and constraints for these two models. The fourth section will describe the optimal results obtained from these model formulations, and discuss key findings. Particular emphasis will be placed on how these

models compare not only to other formulations but also how well these models meet the stated research questions. The fifth and final section will offer concluding remarks and a brief discussion for possible future work.

6.2 Background

As was seen in Chapter 2, many real world transit routes in small to medium sized cities utilize a mix of route alignments, some of which involve embedded loops. The routes/paths that involve a combination of straightway sections and loops often help to extend coverage to nearby areas without the need to use additional transit routes. In some cases this can be attributed to the overall design of the network; for example, it may be that a downtown business district is composed of city blocks with alternating one way streets comprising the grid pattern. In such a case a tour/loop might be a necessity. For example, Figure 6.1 presents a map of transit route 2 used by San Luis Obispo Transit. This particular route uses loops in several sections, however, it should be noted that the two loops in the northern section of the route use several parallel street segments that are separated by several blocks. Two of these streets that are used in this section of the route (i.e. Marsh and Pismo streets) are one way, and the route travels up one street and down the other. Thus, the network layout forces such loops (departures) to be used. However, as several route maps that were presented in Chapter 1 demonstrated, tours may be used in a wide variety of other circumstances and forms. For example, a 'barbell' shaped route as depicted in Figure 6.2a, a 'lollipop' shaped route as depicted in Figure 6.2b, or even a figure eight shaped route as depicted in Figure 6.2c may be used by transit planners in providing transit access coverage.

There may even be combinations of the patterns shown in Figure 6.2 within the same route. Figures 6.3 and 6.4 present system route maps for the Cities of Bozeman, MT and Eau Claire, WI respectively. For the City of Bozeman, four of the five routes involve one or more loops. Note that the "yellow" route contains a large loop at the western end of the route (creating a lollipop shape), where the loop itself is traveled only in the clockwise direction. The "red" route contains two loops and two common sections, where each of the embedded loops is traversed in only one direction. The "blue" route shows the use of a loop in the center of the route, where the two sections of the route are separated by 5 blocks. This clearly represents a pattern that is broadened in order to cover more of the neighborhood with access coverage. Note that the "orange" route is primarily a single loop (with a smaller loop in the Northwest section). Note also that the main loop, as well as the smaller loop, is traversed in only one direction. The system map for the City of Eau Claire depicts a wide variety of route shapes. For example, routes 17 and 18 are lollipop shaped routes and route 3 involves the use of three loops (traveled in only one direction), and several long sections traversed in both directions. Overall, one can see that routes, themselves, are not simple out and back features, but are of a more complex nature that provide greater levels of route coverage while saving some route mileage. When a loop is traversed in only one direction of travel, it can help to extend route coverage at the expense of increasing travel times. This represents a tradeoff that planners are forced to make when housing densities and transit demand is low. The idea is that area coverage can be increased by introducing loops with a lower marginal cost (distance of operation) than using an additional new route. This certainly increases travel times for riders, but at the same time keeps transit route mileage as low as possible. For example,

consider the two simplistic route designs depicted in Figure 6.5; Route A is a simple route, going up and down a street 5 blocks long, whereas route B is a loop route that travels up that same street and then traverses over 3 blocks and travels down that street, eventually traversing back to the original street. Whereas Route A serves 0.6 unique intersections for each block traveled, Route B serves 1 unique intersection for each block traveled, making route B more efficient in terms of spatial coverage per unit mile traveled. Without doubt, customers travel less to move between their respective origins and destinations along Route A as compared to what customers must travel between their respective origins and destinations along Route B. Thus, route coverage is increased at the expense of increasing service times. However, for most areas, it is better to have some level of coverage even when service times are increased as overall access is increased.

As we have seen, models such as Current et. al. (1984, 1985) or a TSP structure such as Vajda (Vajda, 1961; Curtin and Biba, 2011) explicitly prevent loops/tours from being part of a solution unless the entire route is one large TSP loop. This of course stems from the idea that an optimal route will never contain a loop or tour. This assumption holds for a problem such as the simple shortest path problem, but for maximal covering-shortest path problems it does not hold as a general assumption, as demonstrated in chapters 3, 4, and 5 of this dissertation. However, all past work on the maximal covering shortest path problem has been based upon models that represent a route which is determined based upon one direction. The results of these models are then assumed to be reversible; that is, vehicles will retrace their routes or paths in the opposite direction. For example a route with a loop is depicted in the right panel of Figure 6.6, where each segment is traversed in both directions. The left panel of Figure 6.6 depicts

the same alignment with the exception that the route travels the loop in only one direction. Whereas the solution depicted in the right panel of Figure 6.6 can be generated by the models in chapter 3, the solution depicted in the left panel of Figure 6.6 falls outside the scope of existing models.

In the typical planning scenario, routes are often drawn by hand and subsequently tweaked slightly in order to conform to surface street conditions. This is exactly what is done through the Trapeze (www.trapezegroup.com) and remix (formerly TransitMix – www.getremix.com) software packages. Thus, a key goal of this dissertation is to propose and develop a new fundamental model for route planning which accounts for travel in both directions. Further, previous formulations have restricted a loop from forming outright. Chapters 3 and 5 presented models which are 'loop agnostic' in the sense that the model allows a loop to be used, where travel is assumed to take place in both directions. Loops in the previous chapters can be considered to be bi-directional loops; they are bi-directional in the sense that the vehicle traveling the route or pathway traverses all arcs in both directions, including the arcs which form loops. What is assumed here is that a loop, if used, may or may not be traversed in both directions. This chapter will present two models that delineate routes (or pathways) with respect to both directions of travel; that is, inbound and outbound with respect to traveling from or to the origin. The two models presented here are designed to find routes that can involve uni-directional loops, that is, loops which travel in only one direction which are found in real world examples.

Intuitively, it is quite likely that if one solves for a bi-directional path that the inbound and outbound paths will share little, if any, street segments in common. Thus,

two almost independent paths will be generated, except for the notion of maximizing combined coverage. That is, little will be gained in sharing any street segments between the outbound and the inbound direction. Boffey and Narula (1998) seem to have been aware of this fact as they defined their problem in such a way that 2 (or more) paths are determined for the same origin and destination nodes. This in effect is an attempt to model several routes as part of a whole system wherein each route is then assumed to provide complementary coverage to the other routes. The research here differs from that of Boffey and Narula in that paths should not necessarily be independent, but share some of the route, so that travel times can be lower than what might occur in one large loop or two completely independent directional paths. Two forms of this problem were presented in the introduction: a form that requires a certain proportion of route segments to be shared by both travel directions (inbound and outbound) and a form that encourages a certain number of intersections to be shared between the two route directions. The next section will present model formulations for the two versions of this problem.

6.3 Notation and Formulation of the Bi-Directional Covering Path Problem

This section will present the notation and two formulations for the Bi-Directional Maximal Covering Shortest Path Problem. The notation given below will be used for both variations of the BD-MCSP problem. Each formulation is designed such that it is possible to traverse all or only parts of the route in both directions. Any departure in path alignment between the outbound path and the inbound path represents the use of a uni-directional loop. The notation for these models is as follows:

i, j = the index of nodes

r = the index for path direction, where *r=1* refers to the outbound path from origin to destination and *r=2* refers to the inbound path from the destination to the origin.

a_k = population or measure of demand at *k*

α = the importance weight associated with the coverage objective

$\beta = 1 - \alpha$ = the importance weight associated with the distance objective

d_{ij} = the shortest distance/time from node *i* to node *j*

x_{ij}^r = 1 if the arc from *i* to *j* is traversed from *i* to *j* for path direction r

y_k = 1 if node *k* is covered and zero if it is not

node p = the origin (destination) node for the outbound (inbound) bi-directional shortest covering path

node q = the destination (origin) node for the outbound (inbound) bi-directional shortest covering path

$N_j = \{ i | \text{arc}(i, j) \text{ exists } \}$, the set of nodes *i* which are connected to *j*

S^* = the desired maximum allowable access distance

$S_k = \{ j | d_{jk} \leq S^* \}$, the set of nodes *j* which are within the maximum access distance to node *k*

F_i = the set of nodes, *k*, that are connected to *i* by an arc and can be directly traversed from *i* to *k*.

T_j = the set of nodes, *k*, that are connected to *j* by an arc and can be directly traversed from *j* to *k*.

Θ_{ij} = variable to define arc-sharing by both directions; 1, if the outbound path (1) traverses arc(*i*,*j*) from *i* to *j* and the inbound path (2) traverses arc (*i*,*j*) from *j* to *i*; 0, otherwise

Θ_{ji} = variable to define arc-sharing by both directions; 1, if the outbound path 1 traverses arc(*i*,*j*) from *j* to *i* and the inbound path 2 traverses arc (*i*,*j*) from *i* to *j;* 0, otherwise

The first formulation of the BD-MCSP is one that specifies that there must be at least Ω shared arcs that are to be shared among both the forward (outbound) and reverse (inbound) directional paths. This will ensure that the two pathways (inbound and outbound directions) are not completely independent and traverse completely different

arcs and alignments. For example, if an intermediate arc were specified to be used by both directed paths, the model would allow the possibility of a barbell shaped route if such a route was better than other shapes or that the same alignment in both directions for that street segment was better than not. This form of the BDCP problem will be called the Minimum-Shared Arcs Bi-Directional Maximal Covering Path Problem (MSA-BD-MCSP) and is formulated as follows:

$$\text{Maximize} \quad \alpha \sum_i a_i y_i - \beta \sum_i \sum_{j \in F_i} d_{ij} (x_{ij}^1 + x_{ij}^2) \tag{6.1}$$

Subject To

$$\sum_{j \in F_p} x_{pj}^1 - \sum_{i \in T_p} x_{ip}^1 = 1 \text{ for origin node } p \text{ on path 1} \tag{6.2}$$

$$\sum_{j \in F_q} x_{qj}^2 - \sum_{i \in T_q} x_{iq}^2 = 1 \text{ for origin node } q \text{ on path 2} \tag{6.3}$$

$$\sum_{i \in T_q} x_{iq}^1 - \sum_{j \in F_q} x_{qj}^1 = 1 \text{ for destinatio n node } q \text{ on path 1} \tag{6.4}$$

$$\sum_{i \in T_p} x_{ip}^2 - \sum_{j \in F_p} x_{pj}^2 = 1 \text{ for destinatio n node } p \text{ on path 2} \tag{6.5}$$

$$\sum_{i \in T_k} x_{ik}^r - \sum_{j \in F_k} x_{kj}^r = 0, \text{ for } r = 1 \& 2, k \in N \text{ where } k \neq p \text{ and } k \neq q \tag{6.6}$$

$$\Theta_{ij} \leq x_{ij}^1 \text{ amd } \Theta_{ji} \leq x_{ji}^1, \quad \text{for each } i,j \in A \tag{6.7}$$

$$\Theta_{ij} \leq x_{ji}^2 \text{ and } \Theta_{ji} \leq x_{ij}^2, \quad \text{for each } i,j \in A \tag{6.8}$$

$$\Theta_{ij} + \Theta_{ji} \leq 1, \text{ for each } i,j \in A \tag{6.9}$$

$$\sum_i \sum_j \Theta_{ij} \geq \Omega \tag{6.10}$$

$$\sum_{i \in N_j} \sum_{j \in s_k} \left(x_{ij}^1 + x_{ij}^2\right) - y_k \geq 0, \forall\, k, k \neq p, q \tag{6.11}$$

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij}^r - \sum_{i \in V} \sum_{\substack{k \in T_i \\ k \notin V}} x_{ki}^r \leq |V| - 1, \text{ for } r = 1 \& 2, \text{ subset of nodes } V, \text{ when } |V| \geq 2 \tag{6.12}$$

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij}^r - \sum_{i \in V} \sum_{\substack{k \in F_i \\ k \notin V}} x_{ik}^r \leq |V| - 1, \text{ for } r = 1 \& 2, \text{ subset of nodes } V, \text{ when } |V| \geq 2 \tag{6.13}$$

$$x_{ij}^{r} = (0,1), \quad \text{for } r = 1 \& 2, \forall (i,j) \tag{6.14}$$

$$y_k = (0,1), \forall k \tag{6.15}$$

The objective of the model (6.1) is similar to that of the MCSP; an importance weight, α, is placed on the covering objective and an importance weight, $\beta = 1 - \alpha$, is placed on the distance objective. In this case, coverage is defined such that population is maximized while the total combined length of the outbound and inbound paths is minimized. The constraints for the problem are somewhat similar to the NR-MCSP, albeit with appropriate conditions imposed on each path direction. In order to account for the bi-directional nature of the problem, we specify that path (1) corresponds to traveling from an origin node *p* to destination node *q* (outbound) and the second path, path (2) – the inbound path – corresponds to travel from node *p* back to the origin node *q*. Thus, constraint (6.2) ensures that one more arc must be used to depart the origin node *p* than is used to enter it with respect to path 1. In a similar vein, constraint (6.3) requires that one more arc must be used to leave node *q* than is used to enter it with respect to the inbound path 2. Constraints (6.4) and (6.5) are a similar type of constraint but in this case require that one more arc must be used to enter the destination node than what may be used to leave the destination, for each respective path (outbound and inbound). Constraints of type (6.6) are "flow" balance constraints for each path direction. They specify that, for each path direction, if a node that is not the origin or destination is entered one or more times by arcs on a directional path, there must be a an equal number of departures from that node for that direction. Constraints (6.7) and (6.8) are introduced here to define each case in which an arc is used in both directions. Because the objective is to maximize coverage while minimizing path length it is highly likely that each path would

take a route that is completely independent of the other path. Essentially, this means that it is likely a solution to the problem would simply be a large covering tour. However, since we know that real world routes often utilize some of the same street segments in both directions, it is necessary to ensure that each path comes together and shares a number of common arcs which are traversed in the forward direction by one path and the reverse direction by the other. To establish that there is a minimum number of shared arcs, we introduce a decision variable, Θ_{ij}. This decision variable is used to represent whether an arc is shared by both path directions; if both paths utilize the same arc (i, j), but in opposite directions, then this variable (Θ_{ij}) can have a value of one. If only one or neither path use this arc then the variable will be forced to a value of zero. Constraints (6.7) account for travel on the 'forward' portion of the bi-directional path. If the arc is traversed from node *i* to node *j* on path 1 (the 'forward' path) then Θ_{ij} is allowed to have a value of one. If it is traversed in the opposite direction for path 1, then Θ_{ji} is allowed to be one in value. Constraint (6.8) ensures that if the arc is traveled in the 'reverse' direction as part of path two from node *j* to node *i* then Θ_{ij} is allowed to have a value equal to one and Θ_{ji} is allowed to be one in value if the reverse direction travels from *i* to *j*. This ensures that Θ_{ij} (and conversely Θ_{ji}) can only have a value of 1 if the arc is utilized as a part of both paths in the opposite directions. If either path uses an arc which is not utilized in the opposite direction by the other path, then the respective constraints will force Θ_{ij} and Θ_{ji} to have values of 0. Constraints (6.9) are necessary to prevent a case where a shared arc is counted twice. For example, if there is an arc that connects node *i* to node *j* there will be two variables representing the directional travel for each

path (outbound and inbound). That is, we will have the variables x_{ij}^1, x_{ji}^1, x_{ij}^2, and x_{ji}^2 which represent travel from node *i* to node *j* and from node *j* to node *i* for the outbound and inbound paths, respectively. However, because Θ_{ij} and Θ_{ji} exists for each complementary inbound and outbound path arc pair variables (i.e. x_{ij}^1 and x_{ji}^2) it is possible that a single arc segment could be counted twice if x_{ji}^1 and x_{ij}^2 are also utilized as Θ_{ij} and Θ_{ji} would both be equal to 1. This of course violates the fact that we wish to count shared arc use only once for a given arc; constraints (6.9) prevent such double counting by ensuring that only one pair of inbound and outbound arc traverses can be counted toward the minimum number of shared arcs, Ω, as only one pair of directions is allowed to be used in calculating shared directional use. Constraint (6.10) requires that there must be at least Ω arcs that are common to the forward and reverse paths as this defines the minimal amount of arc sharing between the two directions. Since each pair of Θ_{ij} and Θ_{ji} variables can have in common at most a value of 1 if based upon constraints (6.9), we can ensure that there will be at least Ω arcs used in common between the inbound and outbound directions with constraints (6.10). By ensuring that the number of arcs that are used in the opposite directions is equal to or greater than Ω, we allow for the possibility that more than this number of shared arcs could be part of a solution. Constraints of type (6.11) define whether coverage has been provided by either path direction. This constraint ties the coverage variables y_i to the arc traversing variables x_{ij}^r. In this case, the constraint specifies that node *k* can be considered covered if an arc on either path direction visits a node *j* which is within the maximum access distance of

node k. If this requirement is fulfilled, then y_i is allowed to be 1 and node k is considered to be covered.

Constraints (6.12) and (6.13) represent the EAST constraints for the problem. As we have seen, sub-tours could be present in a covering-path solution, and because of this should be allowed to occur only if they are connected to the covering path. These EAST constraints are iteratively added to the problem as described in Chapter 3 of this dissertation; if a sub-tour is identified in a solution then the appropriate EAST constraints are added and the problem is re-solved. In this case, we will need to identify whether sub-tours occur with respect to each path direction and add the corresponding EAST constraint to prevent its use unless it is attached to the path. Constraint (6.12) requires that in order for the loop/tour to exist for a specific path direction, it must be entered by an arc which is not part of the sub-tour. Similarly, constraint (6.13) requires that in order for the loop/tour to exist in a specific path direction, it must be left by an arc for that same path direction which is not part of the sub-tour. Taken together, these constraints will either eliminate or attach specific sub-tours within a solution. Finally, constraints (6.14) and (6.15) represent the binary restrictions on the path and coverage decision variables, respectively.

This formulation for the Minimum Shared Arcs Bi-Directional Maximal Covering Shortest Path problem is able to capture elements that define real world transit routes. In particular, it addresses the issue of defining a route in both directions as this allows for a realistic accounting of coverage and travel. By requiring a minimum number of arcs to be shared between both routes, one can explore solutions which vary in route similarity between the outbound and inbound directions. In particular, this formulation allows for

the use of uni-directed loops as observed in Figures 6.3 and 6.4 for the cities of Bozeman, MT and Eau Claire, WI. An example of an optimal uni-directed loop formed by both an inbound and outbound path for the Swain network is given in Figure 6.7.

The formulation above is not without its drawbacks. Although the problem at a basic level is a combined form of two MCSP problems, the addition of the constraint requiring at least Ω arcs be shared among both path directions introduces a knapsack or budget constrained sub-problem. Budget and Knapsack constraints involving integer variables are often computationally intensive constructs when solving ILP problems to optimality. Unfortunately, although the MSA-BD-MCSP problem is conceptually simple, it is very difficult to solve to optimality for reasonable sized problems; this is due in no small part to the fact that it can be classified as NP-Hard. Due to this fact, it is likely that this and other model forms of the MSA-BD-MCSP will require significant amounts of computer time to solve to optimality.

To address this possibility, an alternate formulation is proposed which utilizes an additional objective term that contains an emphasis weight which is used to encourage an inbound (or outbound) route to visit a node previously visited by the outbound (or inbound) route, rather than requiring them to coincide with at least a minimum number of arcs. This objective term can be specified as:

$$\mu \sum_{i=1}^{n} a_i t_i \qquad \textbf{(6.16)}$$

where μ is the importance weight placed on the benefit to return to a node visited by the opposite directed path, a_i represents the population at node *i* and t_i is the binary decision variable representing whether the return coverage benefit at node *i* is provided or not

provided. This weighted term is used to capture the benefit of serving and visiting a node in both directions. In this case we define this benefit to be some portion of coverage; thus, the emphasis weight can represent the percentage of additional coverage to be added when a node is visited by both directed paths. It should be noted that it is possible to link the return coverage benefit weight to the emphasis weights placed on overall path length and path coverage through the term $\alpha + \beta + \mu = 1$. This allows the weights to be proportional to one another and allows for one to define emphasis placed on each objective term as a percentage. However, we do not require this as we prefer to define the benefit of returning to a node in terms of how much additional coverage can be provided. Using this construct we can define an alternative formulation for the Bi-Directional Covering Path problem which is given below and is called the Weighted Return Bi-Directional Maximal Covering Shortest Path Problem (WR-BD-MCSP). This new model represents a form of the second type of bi-directional path problem defined in the introduction:

$$\text{Maximize} \quad \alpha \sum_i a_i y_i - \beta \sum_i \sum_{j \in F_i} d_{ij} (x_{ij}^1 + x_{ij}^2) + \mu \sum_i a_i t_i \qquad \textbf{(6.17)}$$

Subject To

$$\sum_{j \in F_p} x_{pj}^1 - \sum_{i \in T_p} x_{ip}^1 = 1 \text{ for origin node } p \text{ on path } 1 \qquad \textbf{(6.18)}$$

$$\sum_{j \in F_q} x_{qj}^2 - \sum_{i \in T_q} x_{iq}^2 = 1 \text{ for origin node } q \text{ on path } 2 \qquad \textbf{(6.19)}$$

$$\sum_{i \in T_q} x_{iq}^1 - \sum_{j \in F_q} x_{qj}^1 = 1 \text{ for destination node } q \text{ on path } 1 \qquad \textbf{(6.20)}$$

$$\sum_{i \in T_p} x_{ip}^2 - \sum_{j \in F_p} x_{pj}^2 = 1 \text{ for destination node } p \text{ on path } 2 \qquad \textbf{(6.21)}$$

$$\sum_{i \in T_k} x_{ik}^r - \sum_{j \in F_k} x_{kj}^r = 0, \text{ for } r = 1 \& 2, k \in N \text{ where } k \neq p \text{ and } k \neq q \qquad \textbf{(6.22)}$$

$$\sum_{i \in N_j} \sum_{j \in s_k} \left(x_{ij}^1 + x_{ij}^2 \right) - y_k \geq 0 \,, \forall\, k, k \neq p, q \qquad \textbf{(6.23)}$$

$$t_i \leq y_i \,, \forall\, i, i \neq p, q \qquad \textbf{(6.24)}$$

$$t_i \leq \sum_{j \in T_i} x_{ji}^1 \quad \forall\, i, i \neq p, q \qquad \textbf{(6.25)}$$

$$t_i \leq \sum_{j \in T_i} x_{ji}^2 \quad \forall\, i, i \neq p, q \qquad \textbf{(6.26)}$$

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij}^r - \sum_{i \in V} \sum_{\substack{k \in T_i \\ k \notin V}} x_{ki}^r \leq \left| V \right| - 1, \text{ for } r = 1 \,\&\, 2, \text{ subset of nodes } V, \text{ when } \left| V \right| \geq 2 \qquad \textbf{(6.27)}$$

$$\sum_{i \in V} \sum_{j \in F_i \cap V} x_{ij}^r - \sum_{i \in V} \sum_{\substack{k \in F_i \\ k \notin V}} x_{ik}^r \leq \left| V \right| - 1, \text{ for } r = 1 \,\&\, 2, \text{ subset of nodes } V, \text{ when } \left| V \right| \geq 2 \qquad \textbf{(6.28)}$$

$$x_{ij}^r = (0,1) \,, \text{for each r} = 1 \,\&\, 2, \forall\, (i,j) \qquad \textbf{(6.29)}$$

$$y_k = (0,1) \,, \forall\, k \qquad \textbf{(6.30)}$$

The objective (6.17) maximizes coverage and minimizes distance for both directions of the covering path, just as objective (6.1). The new objective term given in (6.16) is added to the objective (6.17) in order to allow for an additional weighted coverage benefit for returning to a node that has been visited by the opposite direction. Constraints (6.18) and (6.19) are unchanged and specify that the origin node for each path direction must be exited one more time than it is entered. Similarly, constraints (6.20) and (6.21) require that the destination node for each path direction must be entered one more time than it is departed. Taken together these constraints allow loops to be formed at the origin and destination nodes in either direction. Constraints of type (6.22) are flow balance constraints for each path and are also unchanged from the previous formulation. They require that an intermediate node, that is a node which is not the origin or destination, must be departed every time that node is entered for a given path direction (inbound or outbound). Constraints of type (6.23) are the constraints that define coverage: if a node is

visited or within the maximum access distance of a node that is part of the covering path (in either direction) then it can be considered covered. However, we also require a new constraint of type (6.24); constraints of type (6.24) ensure that an additional coverage benefit can be counted only if a node has been visited by both directed paths. If y_i is zero then t_i must also be zero as it must be less than or equal to y_i. Since y_i represents path coverage of node i, a node must be covered a first time in order for any additional benefit, t_i, can be counted. Technically, this is a redundant constraint, when considering constraints (6.25) and (6.26). Constraints of type (6.25) and (6.26) are what link a benefit to being visited by the path in both directions. In this case, in order to provide an incentive to return to a node based upon the opposite direction, t_i is allowed to be positive only when a node is directly visited by both directional paths. Constraint (6.25) ensures that, for each node *i*, t_i must be less than or equal to the sum of the arcs which enter node *i* as part of the 'forward' or outbound covering path (i.e. path 1). If one or more arcs are utilized as part of the forward covering path then t_i will be allowed to one as t_i is limited by constraint (6.25). Constraint (6.26) works in a similar manner but in this case t_i can only be positive if node *i* is directly visited by the 'reverse' or inbound covering path (i.e. path 2). Taken together constraints (6.25) and (6.26) ensure that t_i will only provide a weighted return benefit if node *i* is visited by both the 'forward' and 'reverse' covering paths. Thus, the model is allowed to be 'loop agnostic' with respect to loops which form as part of each bi-directional covering path while also establishing an incentive to encourage each path to 'interact' with one another by visiting nodes in common. Such constraints aid in the fact that planners wish to have routes which utilize

a similar common corridor between each path, but allow for some divergence to cover a wider region which is necessary to provide wider spatial coverage.

Constraints of type (6.27) and (6.28) represent the necessary EAST constraints. As before, these are added when needed. Taken together they specify that a sub-tour of a given path direction must have an arc which enters the sub-tour and an arc which leaves the sub-tour but is not part of the sub-tour itself in order for the tour to exist. When solving the model it is efficient to include the 'simple' form of the EAST constraints in order to prevent simple out-and-back (OAB) sub-tours from occurring as such constraints aid in reducing the number of times the iterative sub-tour identification and constraint adding process must be repeated. Constraints (6.29) and (6.30) are the binary restrictions imposed for the decision variables for path variables and coverage variables respectively. Note, it is not necessary to maintain special restrictions of the t_i variables as they will naturally be either zero or one in value due to the restrictions of the y_i variables and the nature of the constraints (6.25) and (6.26)

These two formulations specified above are designed to find complementary paths, outbound and inbound, that together cover as much as possible while at the same time are efficient in terms of path distances and share either links or visit nodes in common when possible. To address the fact that the classic MCSP models are based upon the assumption that the path can be traveled in both the forward and reverse directions relative to the path alignment, any conflicts based upon network topology must then be modified after the fact in an ad hoc fashion. Both of the model formulations proposed in this chapter do not require such ad hoc adjustments as they can be applied to any

typology, including a mix of one-way streets and two-way streets. The BD-MCSP formulations are 'loop agnostic' like the formulations of the NR-MCSP. However, they allow both bi-directed paths and uni-directed paths to be present in a solution. The first formulation (MSA-BD-MCSP) ensures that a certain number of arcs have to be shared by both the outbound and inbound parts of the path, while the other formulation (WR-BD-MCSP) involved finding complementary path directions (outbound and inbound) which optimized path coverage, minimized path distance, and provided added benefits to visiting nodes in common for the two directions or travel. Both formulations then allow for uni-directional oriented loops (e.g. a loop that travels in either a clockwise or counterclockwise rotation, but not both) to occur which is not possible in the NR-MCSP formulation of the problem developed in Chapter 3. The next section will highlight results for solving both of these models, and discuss nuances that occur for each formulation.

6.4 Network, Computational Environment, and Results

Both formulations above are new, and represent model forms that allow for both uni-directional loops and bi-directional loops. The task at hand is to provide a demonstration of these model constructs, not to develop an efficient, tailored solution process. We applied the models on the 55 node Swain based network dataset that was utilized in the previous chapters of this dissertation. The network is defined to allow travel in both directions along each arc; there are 104 arcs in the dataset which yields a total of 208 directed arcs, and 416 directed path variables.

All problems were formulated and using the Xpress-IVE modeling environment (a product of the FICO Corporation) on a Windows 7 Professional workstation and

solved using the Xpress solver on a Linux based server. The server utilized the Ubuntu 14.04 LTS operating system and the Xpress 64-bit solver v3.8.0. The computer hardware was comprised of two physical Intel Xeon X5560 processors providing a total of 8 logical and 8 virtual cores. This server utilized 48GB of DDR3 memory running at 1066 MHz. The server also utilized five 36GB serial attached storage (SAS) drives in a RAID 5 array which provides 136GB of data storage. All solution information was directed to a file that was uploaded to a dropbox folder that was synced to the Windows 7 Pro workstation. This final step allowed all bi-directional paths to be displayed as a map using a program developed in Microsoft's Visual Studio 2010 VisualBasic.Net programming environment.

The first solution set involves results with respect to solving the MSA-BD-MCSP. The MSA-BD-MCSP was solved for a series of coverage and distance weights as well as Ω, the minimum number of required shared arcs (i.e. arcs shared in both the outbound and inbound path directions). A more exhaustive set of solutions such as provided in Chapter 3 and given in Appendix A is not included here as determining optimal solutions to certain cases of the MSA-BD-MCSP proved to be so computationally complex that determining a confirmed optimal solution to some problems was not feasible – in one case a single problem instance took more than 2 weeks to solve before the decision was made to terminate it. Solutions for selected MSA-BD-MCSP problems are given below in Table 6.1. To generate this set of results, cover weights were incremented by units of 0.1 with weights ranging from 0.1 to .90 for coverage and path emphasis.

Table 6.1 - Solution Results for the MSA-BD-MCSP with p = 27 and q = 21

Cover Weight	Distance Weight	Omega Value	Service Distance	Total Covered	Total Length	Objective Value[18]	Time (Sec)
0.01	0.99	9	10	537	103.84	103.832	0.09
0.10	0.90	9	10	537	103.84	103.756	0.19
0.20	0.80	9	10	552	106.40	102.720	2.29
0.30	0.70	9	10	563	110.73	100.611	5.52
0.40	0.60	9	10	592	124.51	93.906	65.22
0.50	0.50	9	10	620	144.33	82.165	133.01
0.60	0.40	9	10	620	144.33	69.732	823.64
0.70	0.30	9	10	633	166.06	54.718	1503.99
0.80	0.20	9	10	638	184.89	38.578	29389.50
0.90	0.10	9	10	640	196.55	19.655	6478.77
0.99	0.01	9	10	640	196.55	1.966	5728.44

In order to exclude potentially inferior solutions (i.e. multiplying an objective by zero) weights of zero and one were approximated through the use of 0.01 and 0.99 respectively. This results in a total of 11 problems detailed in Table 6.1. The results here were generated for a maximum access distance of 10. Total path length is reported which represents the combined distances of the inbound and outbound directions. The combined coverage for the inbound and outbound portions of the path is also given for each problem. If the distance required to travel through the network for each node j was less than the maximum access distance, then that node is considered to be in the coverage set of node j. All calculations utilized an Ω shared arc value of 9. The table also gives the composite objective value and computational time for each problem. It should be noted that in this case the objective is given as a combination of weighted terms of minimizing what is not covered and minimizing total bi-directional path distance. Shaded rows denote solutions that involve loops, such as a uni-directional loop and/or bi-directional loop.

[18] Note that the objective minimizes total bi-directional path distance and what is not covered.

Figure 6.8 is an example of the optimal solution given in the table above that utilizes an Ω minimum shared arc value of 9, a maximum service distance of 10, and coverage and distance weights that are equal to 0.7 and 0.3 respectively. The outbound portion of this solution from node *p* to node *q*, in this case from node 27 to 21, utilizes an out-and-back loop from node 43 to node 55 and then back to node 43. This out and back loop is a portion of a unidirectional section from node 16 to node 43 and from 43 to node 42. Note that the return direction from node 21 to node 27 includes a different return route from node 42 to node 16, thereby forming a larger uni-directional loop. Thus, several loop structures occurred in this specific bi-directional solution. This also demonstrates that an out and back loop can be attached to a uni-directional section. Note that the formulation requires that each directional path share at least Ω segments. Since the loop from node 43 to node 55 and back to node 43 is formed as a section of only one of the directed paths, it is not counted as a shared arc in both path directions. Its existence in the solution demonstrates that the "loop agnostic" nature can be an advantageous element in the bi-directional problem as well.

Figure 6.9 presents an interesting case that highlights the use of a single uni-directional loop as an optimal solution strategy, particularly when higher importance is given to combined shorter path distance. This solution was generated using an Ω minimum shared arc value of 9, a maximum service distance of 10, and coverage and distance weights that are equal to 0.3 and 0.7 respectively. The objective value, solution time, and path length are listed in Table 6.1 for this solution. What is unique in the solution, however, is that a single uni-directional loop is formed utilizing the outbound path to form one half of the loop and the in-bound path to form the other portion. In this

case, the outbound path traverses from node 3 to node 8 to node 2 and the inbound path traverses from node 2 to node 7 to node 3. This results in the formation of a clockwise uni-directional loop. It should be noted that an alternate optimal solution could be the formation of a counterclockwise loop if the inbound and outbound path segments were reversed in terms of direction. That is, an alternate optimal solution could comprise an outbound path traversing from node 3 to node 7, and from node 7 to node 2 and an inbound path that traverses from node 2 to node 8, and from node 8 to node 3. The key point is that an optimal solution to this problem utilizes a uni-directional loop, which is one of the key contributions of this dissertation.

As was mentioned above, we were unable to generate an extensive set of optimal solutions due to the computational complexity of the MSA-BD-MCSP model. One of the solutions that was generated and not listed in Table 6.1 involved a maximum service distance of 7.5, an Ω value of 11, a distance weight of 0.15 and a coverage weight of 0.85. The reason for showing this solution is that it contains several looped structures that were discussed at the beginning of this chapter and which can be found on real world transit systems. In particular, there is a uni-directional loop formed by each half of the MSA-BD-MCSP path. In this case, the loop is formed from node *p* (equal to node 27) through the graph to node 1 on the outbound path and from node 1 through the graph back to node *p* on the inbound path. This particular version of the uni-directional loop travels in a clockwise direction, although as noted above the respective outbound and inbound path segments could be swapped in direction to form an alternate optimal solution that utilizes a counterclockwise uni-directional loop. This solution also utilizes a bi-directional loop comprised of nodes 1, 5, 11, and 13. In this case, the loop is

traversed in one direction by the inbound and in the other direction by the outbound paths, which results in a bi-directional loop. It is important to note that a common path also exists between nodes 1 and 21. Thus, Figure 6.10 illustrates nicely the fact that loops can be used as an optimal covering strategy in several different ways which matches what we find in cities in the real world such as Bozeman, MT; Eau Claire, WI; San Luis Obispo, CA; etc. What is important to note is that loops can clearly be an optimal feature, and these features mimic route design that is found in real world city networks.

As was noted in the discussion of computational issues above, the MSA-BD-MCSP formulation is a computationally hard problem that is made more complex through the use of the knapsack like requirement that at least a certain number of arcs must be shared. Although we were able to solve this formulation to optimality for certain parameters with respect to the coverage weight, distance weight, and Ω value, some solutions took a great deal of time and as one increased the value of Ω from moderate values, the problem became essentially unsolvable within a realistic amount of time. As such, we developed the second formulation for the problem, the weighted return bi-directional maximal covering shortest path (WR-BD-MCSP). Since this formulation does not utilize knapsack like constraints, but instead utilizes an objective weight that allows for an additional coverage benefit to encourage a node to be visited a second time, the computational complexity is not as onerous as the MSA-BD-MCSP formulation. Table 6.2 presents solutions to the WR-BD-MCSP for the following parameters. We defined the weighted return objective weight μ in (6.15) to be equal to 0.1, 0.25, and 0.5 as these values seem to intuitively reflect the fact that if a node is visited by the opposite

direction as well, then service to that node is more direct and travel times to and from that node is improved. That is, when a node is visited by both directions, service to that node is better than if it is only visited with either the outbound or inbound directed path. We also set these values to be on the conservative side; that is, the return benefit importance weight is never more than half of what it would be for the first visit. Although we calculated solutions for three different return importance weight values, one would want to solve the problem for a range of return weights based upon known patterns of transit usage and demand as this may generate results that better match regional preferences. Since our network is a 'proof-of-concept' design, the use of a conservative value seems to be a reasonable approach, given that we wish to show that we can replicate real world routing patterns while solving the mathematical model to optimality. Table 6.2 below shows results for a set of solutions to the WR-BD-MCSP problem formulation.

Table 6.2 - Solution Results for the WR-BD-MCSP with p = 27 and q = 21

Cover Weight	Distance Weight	Return Weight	Service Distance	WR Benefit[19]	Total Covered	Total Length	Objective Value[20]	Time (Sec)
0.01	0.99	0.10	10	24.50	545	108.00	-76.970	0.12
0.10	0.90	0.10	10	24.50	545	108.00	-18.200	0.57
0.20	0.80	0.10	10	24.50	545	108.00	47.100	2.19
0.30	0.70	0.10	10	32.60	592	135.39	115.427	5.87
0.40	0.60	0.10	10	32.60	592	135.39	188.166	28.60
0.50	0.50	0.10	10	30.50	620	153.27	263.865	195.13
0.60	0.40	0.10	10	39.10	622	174.21	342.616	6430.26
0.70	0.30	0.10	10	40.00	635	197.39	425.283	1550.06
0.80	0.20	0.10	10	41.10	635	202.30	508.640	31755.90
0.90	0.10	0.10	10	47.60	640	269.18	596.682	13910.40
0.99	0.01	0.10	10	64.00	640	632.14	691.279	27.17
0.01	0.99	0.25	10	93.00	553	130.14	-30.309	6.93
0.10	0.90	0.25	10	100.50	555	137.88	31.908	8.13

[19] This is the benefit provided based upon a node being visited by both the inbound and outbound paths

[20] Note that the objective minimizes total bi-directional path distance and maximizes what is covered as well as the weighted return.

Cover Weight	Distance Weight	Return Weight	Service Distance	WR Benefit[19]	Total Covered	Total Length	Objective Value[20]	Time (Sec)
0.20	0.80	0.25	10	100.50	555	137.88	101.196	15.50
0.30	0.70	0.25	10	100.50	555	137.88	170.484	1168.41
0.40	0.60	0.25	10	112.25	606	178.89	247.316	71.04
0.50	0.50	0.25	10	112.25	622	194.28	326.110	130.53
0.60	0.40	0.25	10	114.75	622	200.53	407.738	511.87
0.70	0.30	0.25	10	117.00	635	235.47	490.859	11934.10
0.80	0.20	0.25	10	123.50	635	259.88	579.524	49.70
0.90	0.10	0.25	10	151.75	640	485.05	679.245	57.12
0.99	0.01	0.25	10	137.50	640	802.32	763.077	383.83
0.01	0.99	0.50	10	201.00	555	137.88	70.049	10.08
0.10	0.90	0.50	10	201.00	555	137.88	132.408	21.31
0.20	0.80	0.50	10	224.50	594	172.64	205.188	19.88
0.30	0.70	0.50	10	224.50	594	172.64	281.852	20.07
0.40	0.60	0.50	10	229.50	606	185.14	360.816	28.55
0.50	0.50	0.50	10	238.00	622	215.92	441.040	480.42
0.60	0.40	0.50	10	247.00	625	238.80	526.480	101.71
0.70	0.30	0.50	10	265.00	629	296.29	616.413	1213.16
0.80	0.20	0.50	10	302.00	635	450.41	719.918	20.98
0.90	0.10	0.50	10	312.50	640	564.89	832.011	73.60
0.99	0.01	0.50	10	295.00	640	785.40	920.746	161.24

The shaded rows of the table indicate the presence of a uni-directional and/or bi-directional loop in the optimal solution. The table includes a column indicating the importance weight for coverage, the importance weight for distance, the importance weight associated with μ (shown in equation 6.16), and the maximum desired access distance. We also highlight the population that is covered by the bi-directional paths as well as the total length of the overall route (i.e. the total distance of both the inbound and outbound paths), and the weighted return benefit for both paths. Also given are the composite objective values for each solution as well as the solution time that was taken to solve the problem. To generate this set of results, cover weights were incremented by units of 0.1 with weights ranging from 0.1 to .90 for coverage and path emphasis. In order to exclude potentially inferior solutions (i.e. multiplying an objective by a zero

weight) weights of zero and one were approximated by the values of 0.01 and 0.99 respectively. As mentioned above, we utilized three unique μ values (0.10, 0.25, and 0.50) to vary the importance weight for returning to a previously visited node. This meant that a total of 33 problems were solved and detailed in Table 6.2. All results given in the table were generated for a maximum service access distance of 10, and all computational tests were performed on the same system utilized to generate the results found in Table 6.1.

Figure 6.11 highlights the solution associated with a cover weight of 0.6, a distance weight of 0.4 and a return weight of 0.1. Note the use of a large uni-directional loop between nodes 16 and 3 of the outbound path and nodes 3 and 16 on the inbound path as well as a shared bi-directional loop between nodes 1, 13, 11, and 5. Figure 6.12 highlights a solution to the WR-BD-MCSP utilizing a cover importance weight of 0.4, a distance importance weight of 0.6 and a return weight of 0.25. In this case, a uni-directional loop is formed between nodes 31, 29, and 18 and a bi-directional loop is formed between nodes 1, 13, 11, and 5. A closer view of the results in Table 6.2 reveals that bi-directional path solutions exist which have a slightly longer total distance and the same coverage, but where an additional visit is provided. This is captured in the weighted return benefit value which is also given in Table 6.2. An example of such a case where total coverage remains the same but an additional return benefit is used in one solution versus another is reflected in the solutions where μ is equal to 0.25 and where the coverage importance weights are 0.5 and 0.6 in Table 6.2. The total population covered for each solution is the same although in one solution a slightly longer path is utilized as this yields a greater return weight value. The reason such a solution occurs is

that the value of return coverage is high enough to change alignments and encourage visiting high values nodes a second time. These two bi-directional path solutions are shown in Figures 6.13 and 6.14. Figure 6.13 depicts the optimal solution to the WR-BD-MCSP problem where node p = 27 and node q = 21. In this case, the emphasis weight on coverage is 0.5, and the importance weight on path length is 0.5; note that a uni-directional loop between nodes 31, 18, 36, 15, 7, and 31 as well as a bi-directional loop between nodes 1, 13, 11, 5 and 1 is used. Figure 6.14 highlights the optimal solution to the WR-BD-MCSP problem where the importance weight on coverage is 0.6, and the importance weight on path length is 0.4. Note that both of these solutions cover the exact same population; the difference is that this solution employs a similar uni-directional loop between nodes 31, 18, 36, 15, 7, and 31 as well as an additional uni-directional loop between nodes 31, 18, 29, and 31. Both solutions also utilize a bi-directional loop between nodes 1, 13, 11, 5, and 1. The distinguishing feature between these two solutions is that an additional node (18) has been revisited. Thus, we can see that the return weight, μ, plays an important role with respect to encouraging the outbound and inbound paths to coincide while eliminating the restriction that Ω minimum arcs must be shared as in the MSA-BD-MCSP model.

One of the interesting results of the model was that we anticipated generating a number of 'figure 8' type routes due to the nature of the formulation. That is, we expected that each path would 'zig-zag' across one another as such a possibility is not expressively prevented in the formulation. However, in our experience this tended not to be the case with one exception: when a very low importance weight is placed on combined overall length. One of the reasons for this seems to be the fact that we utilized

conservative values for μ. This seems to be corroborated by the fact that low μ values reduced such behavior while as μ values were increased 'zig-zag' pathways began to occur at distance importance weights around 0.1 as opposed to importance weights of 0.01. Figures 6.15, 6.16, and 6.17 all show 'zig-zag' solutions which occurred at a distance importance weight of 0.01 and the three μ values of 0.10, 0.25, and 0.50 respectively. This clearly shows that a range of return importance weights should be used as well as a thorough range of coverage and distance importance weights. However, the important thing to note is that for the vast majority of solutions, the inbound and outbound paths shared a large number of arcs, which is precisely what the formulation encourages. Further, this mirrors what we would expect to find on real world routes. The other advantage the WR-BD-MCSP model has with respect to the MSA-BD-MCSP formulation is that there are no knapsack type constraints that must be used. This means that, in general, the WR-BD-MCSP needed less time to converge to optimality for most problems. The other significant advantage is that there were no cases of the WR-BD-MCSP model which did not converge to an optimal solution, unlike in the MSA-BD-MCSP. Although the MSA-BD-MCSP allow one to actively control the minimum number of arcs that must be shared and can be used to find un-supported non-dominating solutions, such solutions are likely to be ignored when compared to the supporting non-dominating solutions. Thus, solutions to the WR-BD-MCSP seem very promising as this would allow a planner to derive optimal routes, and compare those to what are currently in existence or proposed.

The key question of course is whether one could interpret such models to be an inefficient routing strategy or whether looped routes provide good service. However, as

was noted at the beginning of this chapter, it is clear that a route that adheres as closely as possible to the shortest path will result in much shorter transit travel times with respect to the origin and destination and the intermediate points served by the route. However, it is important to highlight the fact that we can define solutions on the tradeoff curve between route efficiency and coverage, while encouraging the sharing of a common corridor of arcs. That is, these two models are capable of minimizing overall path length by diverging in one or both directions while enabling service to areas that may not otherwise be covered. As we have shown, such routes are clearly utilized by transit agencies as looped structures are not uncommon among transit lines. For example, if one consults the Bozeman, MT transit map (Figure 6.3), the Eau Claire, WI transit map (Figure 6.4), or virtually all other transit route maps (especially medium sized cities), one can see that routes often diverge directionally. In Santa Barbara, for example, line 14 has a good portion of the route served in both directions, but also has a sizable uni-directional loop. The key point, of course, is that many real-world systems are likely to utilize both bi-directional and/or uni-directional loop structures. One possible advantage that we have not looked into (which is an exciting opportunity for future work) is that such loop structures could provide a means to increase the number of busses that pass through a geographic region, particularly in areas that have high demand through a central corridor. This could be important as it would effectively increase the number of times the corridor is served and thus reduce wait times and increase the efficiency with respect to time that it would take to go from one point to another along a main corridor or in a central area. The Eau Claire system map in Figure 6.4 readily shows the use of loop structures and corridors and we expect demand along these corridors to be high. Ultimately, the BD-

MCSP formulations are able to form routes that replicate behavior found in real world networks and are able to capture travel in both directions when solving for an optimal covering path. These new models, the MSA-BD-MCSP and the WR-BD-MCSP, now provide an underlying analytical framework that can truly capture a more realistic approach to optimize route design.

6.5 Concluding Remarks and Future Work

We have seen that previous modeling work has not truly addressed the flexibility that transit planners take in designing transit routes. We have also seen that previous models treat travel direction in a transit planning context somewhat as an afterthought. That is models typically determine a covering path alignment in a single direction. These routes are then often applied to transit design with the assumption that travel can simply be reversed along the path and 'tweaked' as needed with respect to alignments such as one-way streets, etc. All models, whether based on Current et. al. (1985), or upon a Vajda (1961) framework such as Curtin and Biba (2011) have been formulated with the implicit assumption that coverage must is provided for each direction of travel, even when this is not the case in actual systems. Even the model formulation of Boffey and Narula (1998), which is a multi-path covering problem, does not truly consider the aspect that a covering path in transit planning should be defined in terms of each direction.

This chapter proposed two bi-directional maximal covering shortest path formulations in an effort to capture the inherent tradeoff between coverage and overall path distance while relaxing the assumption that the inbound path is merely the reverse of the outbound path. The first model that was introduced, the Minimum Shared Arc Bi-Directional Maximal Covering Shortest Path formulation (MSA-BD-MCSP) specified the

minimum number of arcs that should be common to both paths with respect to bi-directional travel. This formulation expanded on the work of Chapter 3 and allowed bi-directional and uni-directional loops to be used if they improved the overall objective. The model is able to account for travel in an 'inbound' and an 'outbound' direction, be loop agnostic through the use of EAST constraints, and allow each path direction to diverge from each other, creating uni-directional loops. In particular, the formulation allows for the formation of both uni-directional and bi-directional loops. We also described one of the drawbacks of this formulation, which is the use of knapsack like constraints requiring a minimum number of arcs to be shared. This prompted the formulation of the Weighted Return, Bi-Directional Maximal Covering Shortest Path formulation (WR-BD-MCSP).

The WR-BD-MCSP formulation eliminates the knapsack requirements for sharing arcs and instead allows for a benefit to be awarded if a node initially visited by the outbound path is visited again by the inbound path and vice-versa. We showed that optimal results to this formulation included the use of shared arcs and mimicked routes seen in real world systems such as Bozeman, MT (Figure 6.3) and Eau Claire, WI (Figure 6.4). We also demonstrated that the return weight can be adjusted and that this value can be solved over several values which result in varying degrees of bi-directional service (service provided on both outbound and inbound paths). We also noted the correlation between the use of 'zig-zag' figure eight routes with higher return weights and very low distance importance weights. Overall, however, we show that both formulations, the MSA-BD-MCSP as well as the WR-BD-MCSP are able to determine optimal solutions which reflect real world conditions as well as account for travel in both directions and use

uni-directional and bi-directional loop features. We also showed that the WR-BD-MCSP appears to be an easier problem to solve, and optimal results can be determined via commercial off the shelf solvers.

Future work should focus on several areas. One of the areas that makes sense to explore is to solve these formulations on a real-world network using known transit data. This would allow us to extend our proof-of-concept work into a true spatial analysis modeling tool. An additional area for future work would explore models which route many bi-directional paths simultaneously. Other avenues of potential work include the use of a weighted benefit associated with traversing an arc in the opposite direction as opposed to visiting a node in each direction such as was done in the WR-BD-MCSP. Such a format may prove useful, particularly as this may allow one to consider both arc and node service values as was used for the TRANSMax II formulation proposed in Chapter 5 of this dissertation. However, although these avenues offer exciting avenues for potential work, the greatest contribution of this chapter lies in the fact that a new methodology for modeling transit routes has been offered. We are now able to model covering paths which travel in opposing directions, allow for the use of uni-directional and bi-directional loops if they improve the objective, and more importantly capture the use of shared arcs and more complex loop features through two new modeling frameworks.

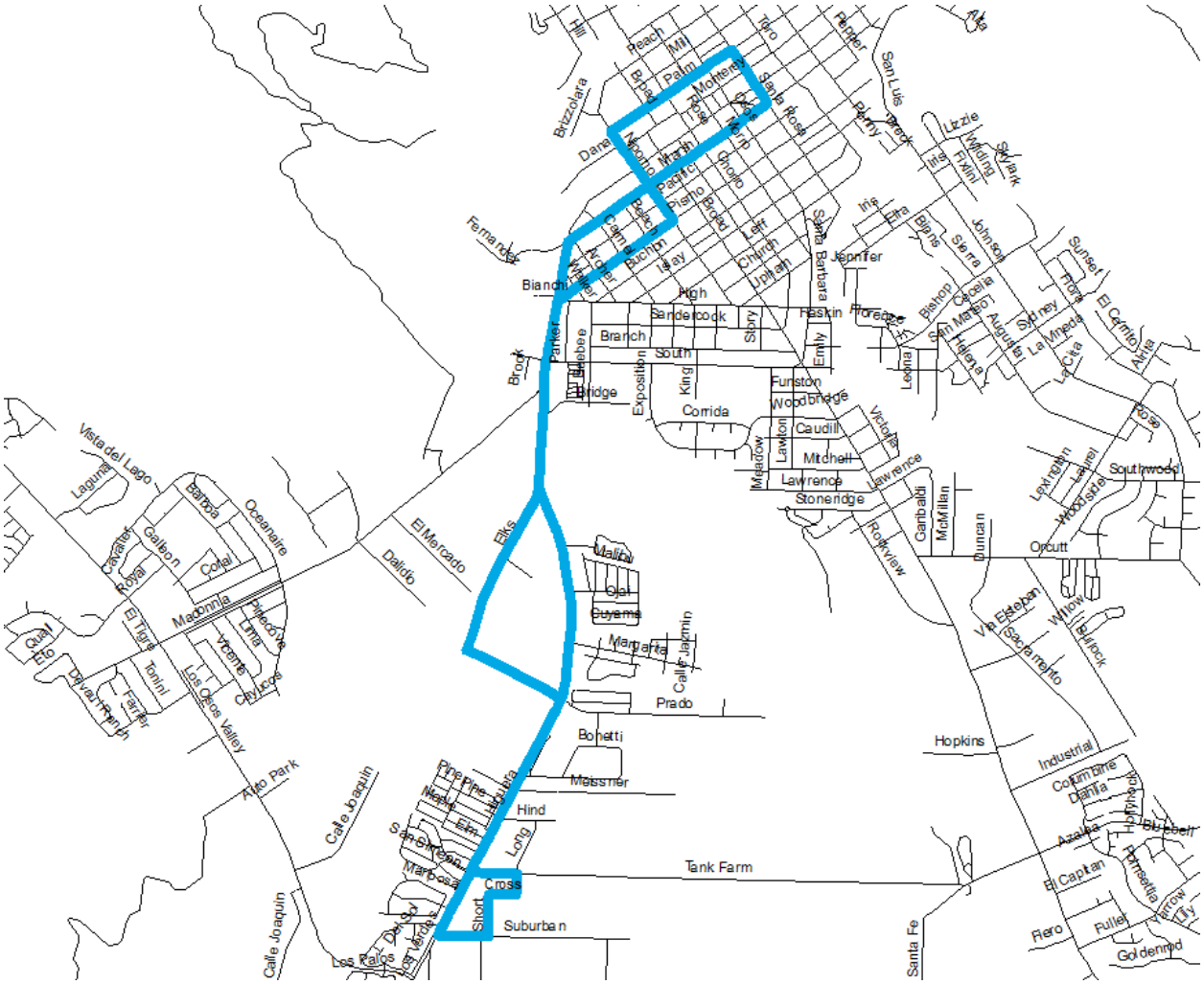
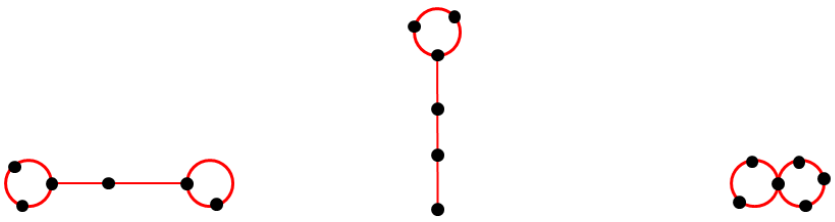


Figure 6.1 - Route 2 in San Luis Obispo, CA which highlights the use of several 'loop' features such as the lollipop, barbell, and figure eight.



A. The Barbell B. The Lollipop C. The Figure 8

Figure 6.2 - Generalized Patterns of Common Tours Formed on Transit Routes

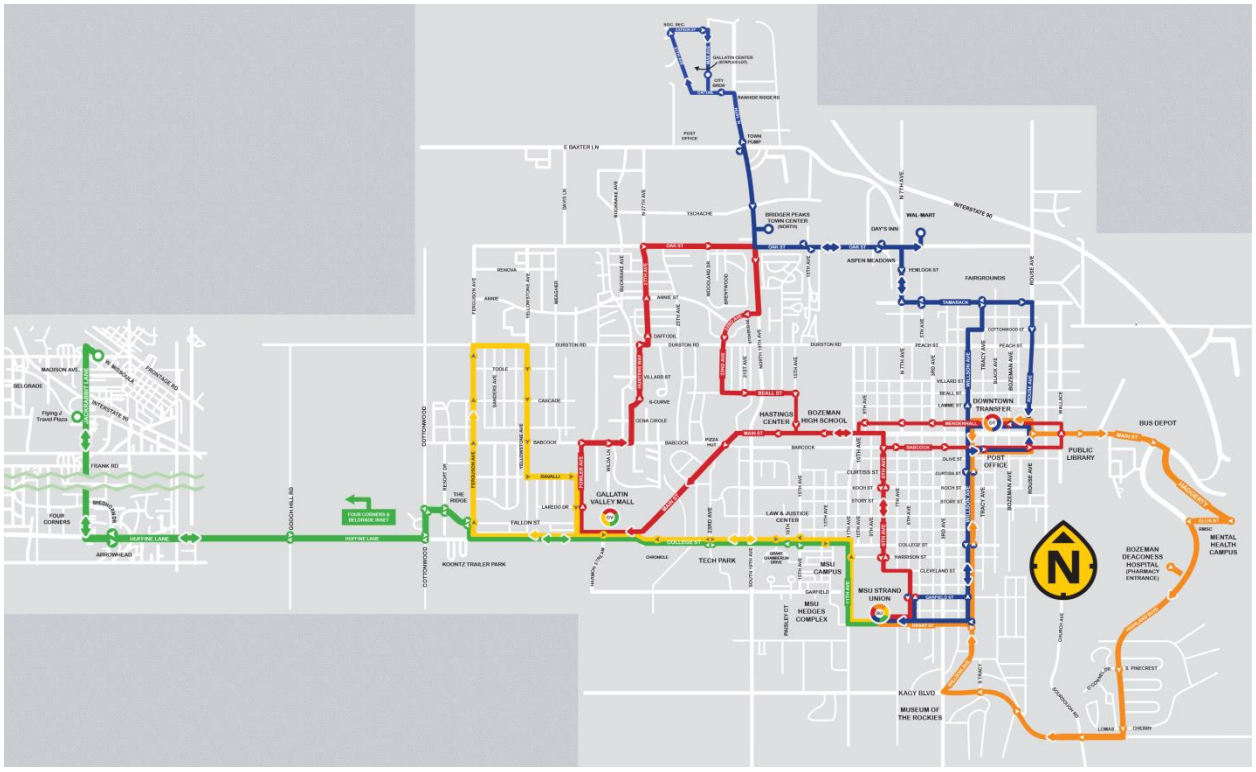


Figure 6.3 - The system route map for the city of Bozeman, MT. Note the use of several loops throughout the system. Permission to publish this route map has been generously given by the City of Bozeman and is accessible at their website http://streamlinebus.com/routes-schedules/route-maps/weekday-service/

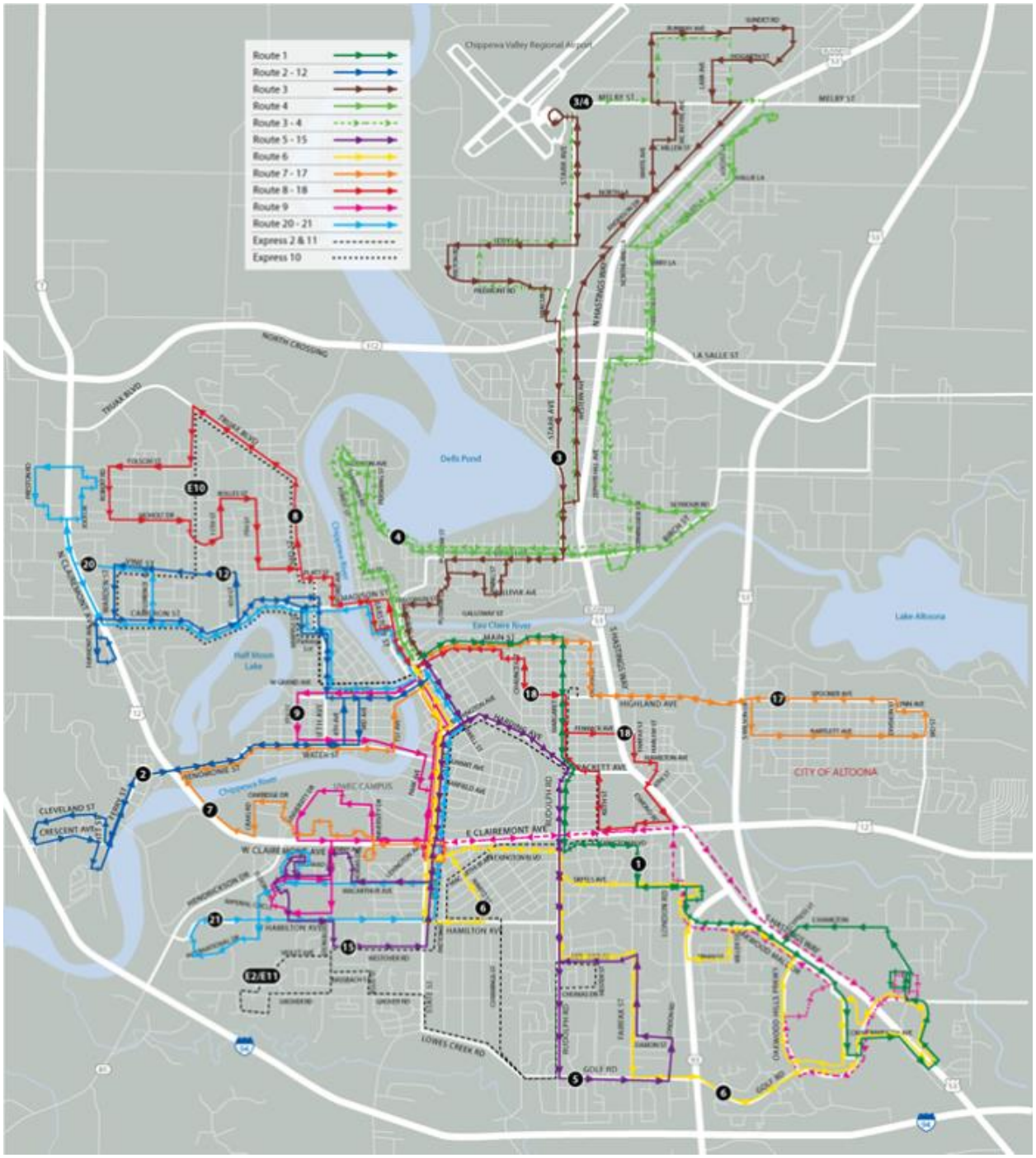


Figure 6.4 - The system route map for the city of Eau Claire, WI. Note the use of many different kinds of loops throughout the system. Permission to publish this route map has been generously given by the City of Eau Claire and is accessible at their website http://www.eauclairewi.gov/departments/transit/maps-schedules

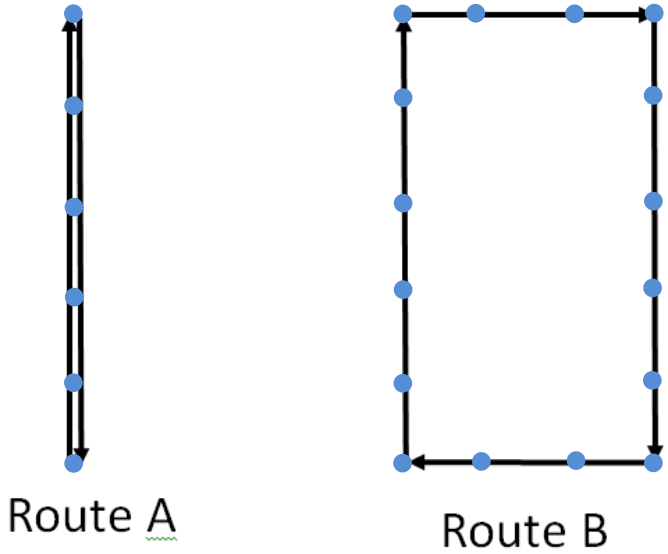


Figure 6.5 - Example of two routes, one out and back using the same alignment (Route A) and the other a loop (Route B). On a grid of 5 by 3 blocks, route A serves 6 intersections (and 6 street segments) and travels 10 blocks. Route B travels 16 blocks and serves 16 intersections (and 16 street segments). Route A serves .6 unique intersections per block traveled and route B serves 1 unique intersection of every block traveled.

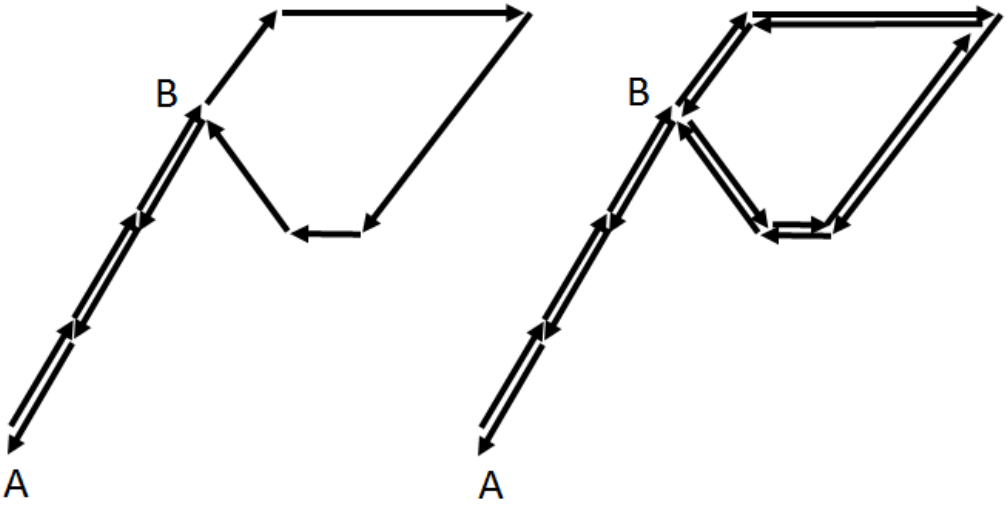


Figure 6.6 - Two routes, each with one loop. The route on the left traverses the loop in only one direction, whereas the route on the right traverses the loop in both directions.

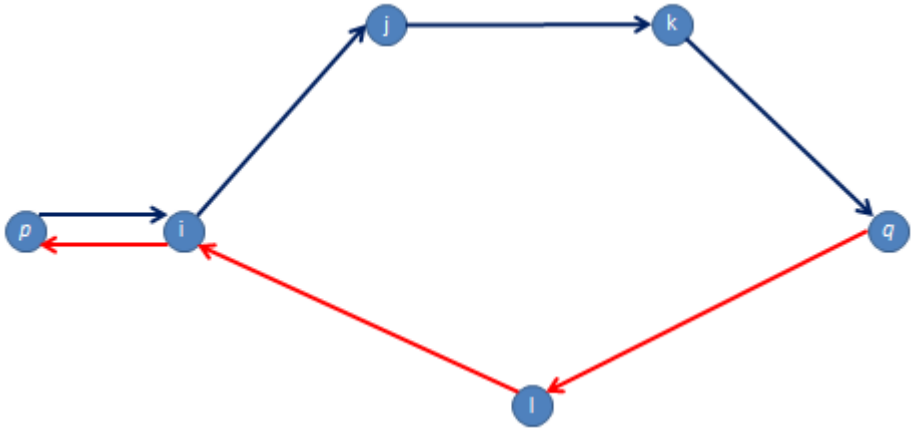


Figure 6.7 - An example of a direction oriented loop utilizing nodes q→l→i→j→k→q formed by a forward path traveling from node p to node i to node j to node k to node q and a reverse path from node q to node l to node i to node p.

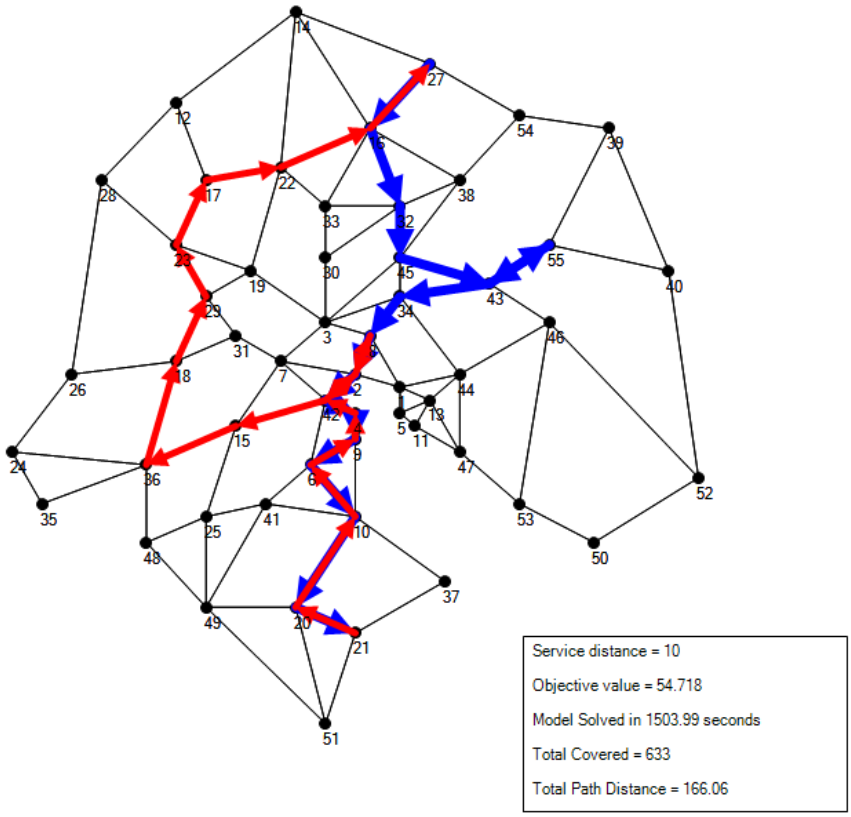


Figure 6.8 - The optimal MSA-BD-MCSP solution for p = 27 and node q = 21, Ω = 9, a maximum service distance = 10, a distance weight = 0.3 and a coverage weight = 0.7. Note the formation of several distinct loop structures.

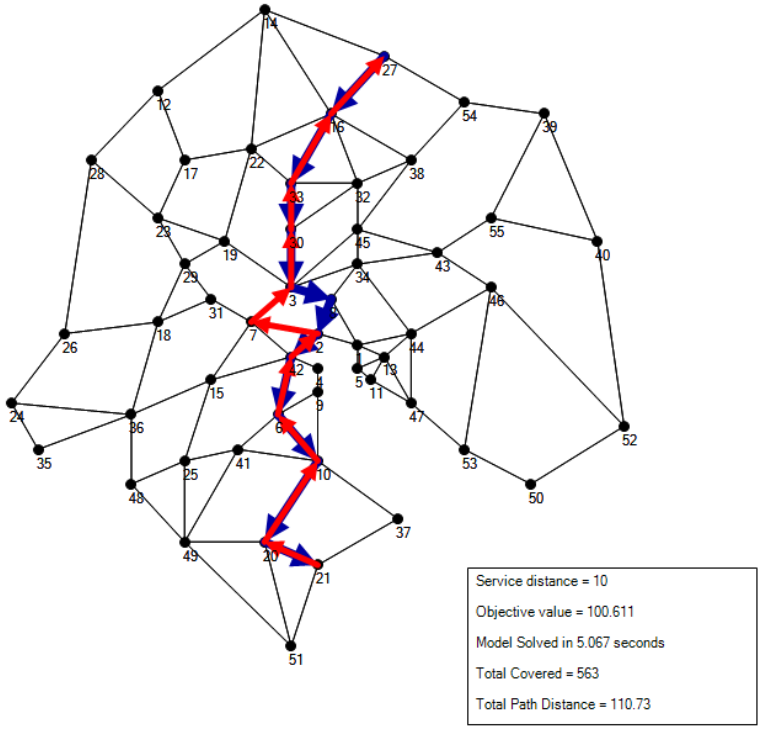


Figure 6.9 - the optimal MSA-BD-MCSP solution for p = 27 and node q = 21, Ω = 9, a maximum service distance = 10, a distance weight = 0.7 and a coverage weight = 0.3. Note the use of a direction oriented loop in the middle of the route.

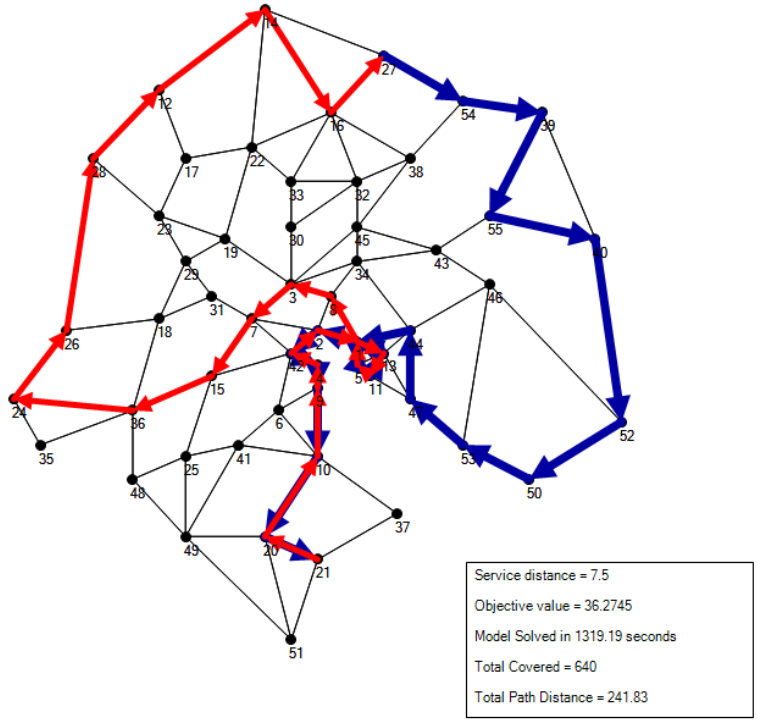


Figure 6.10 - An optimal solution to the MSA-BD-MCSP problem where node p = 27 and node q = 21. In this case a maximum service distance of 7.5 is used, Ω = 11, the emphasis weight on coverage = 0.85, and the emphasis on path length = 0.15. Note the use of a very large uni-directional loop between nodes 1 and 27 of the inbound and outbound paths as well as a bi-directional loop between nodes 1, 13, 11, and 5.

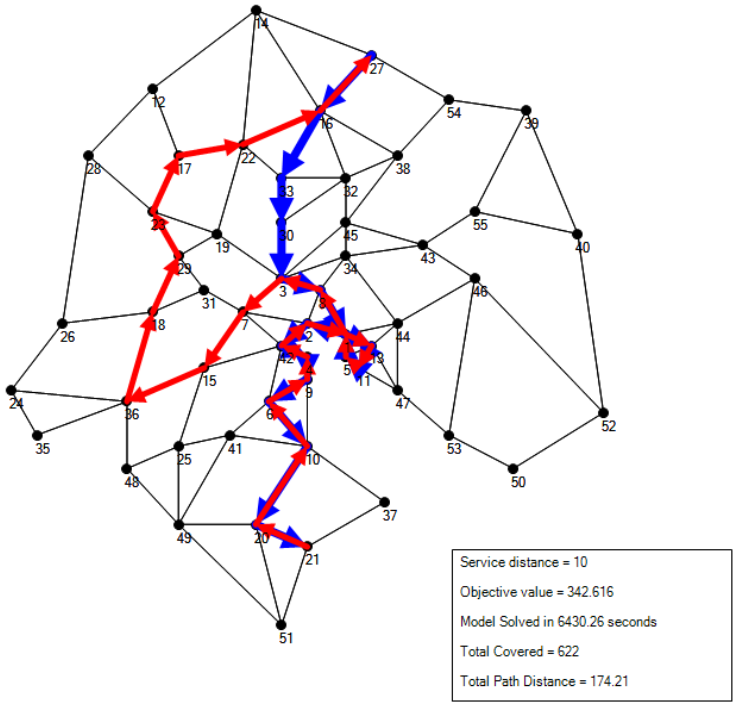


Figure 6.11 – An optimal solution to the WR-BD-MCSP problem where node *p* = 27 and node *q* = 21. In this case a maximum service distance of 10 is used, the emphasis weight on coverage = 0.6, and the emphasis on path length = 0.4. Note the use of a large uni-directional loop between nodes 16 and 3 of the outbound path and nodes 3 and 16 on the inbound path as well as a shared bi-directional loop between nodes 1, 13, 11, and 5.

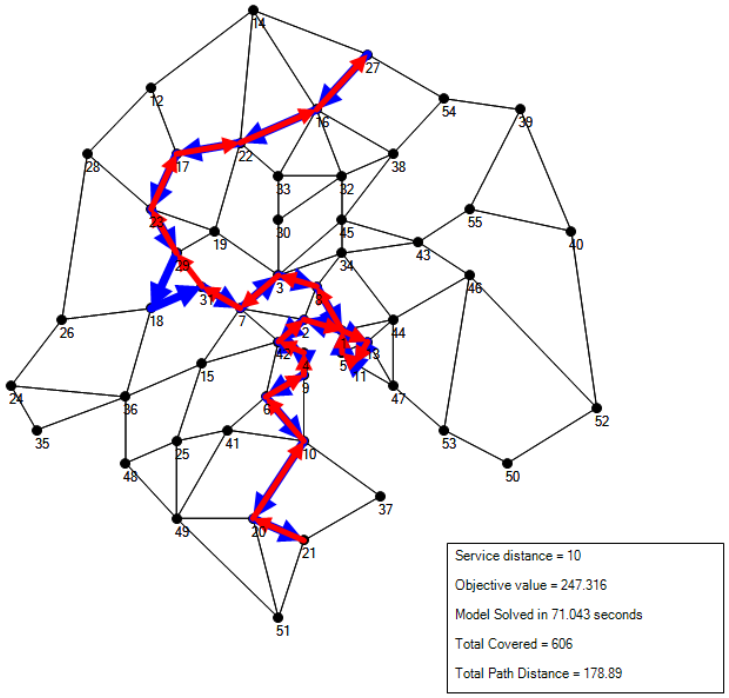


Figure 6.12 – An optimal solution to the WR-BD-MCSP problem where node *p* = 27 and node *q* = 21. In this case a maximum service distance of 10 is used, the emphasis weight on coverage = 0.4, and the emphasis on path length = 0.6. Note the use of a large uni-directional loop between nodes 31, 29, and 18 as well as a shared bi-directional loop between nodes 1, 13, 11, and 5.

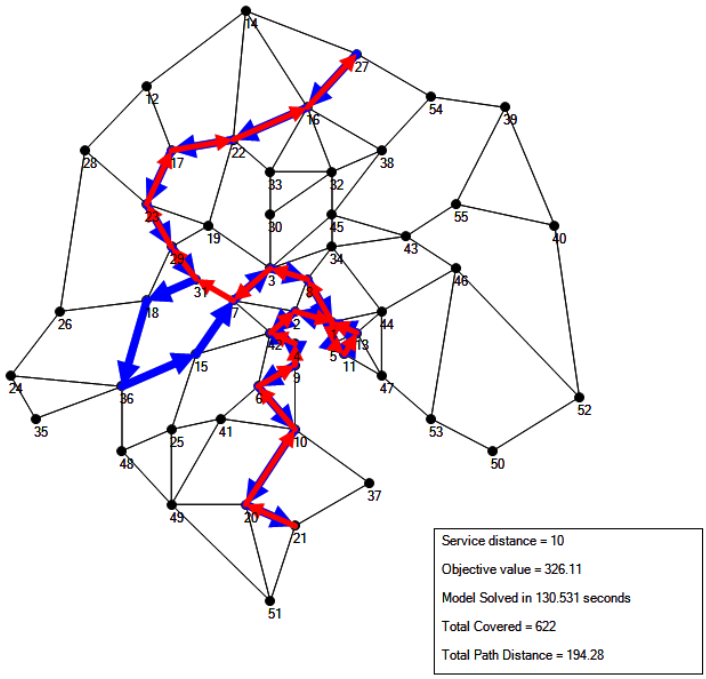


Figure 6.13 – An optimal solution to the WR-BD-MCSP problem where node *p* = 27 and node *q* = 21. In this case a maximum service distance of 10 is used, the emphasis weight on coverage = 0.5, and the emphasis on path length = 0.5. Note the use of a uni-directional loop between nodes 31, 18, 36, 15, and 7 as well as a bi-directional loop between nodes 1, 13, 11, and 5.

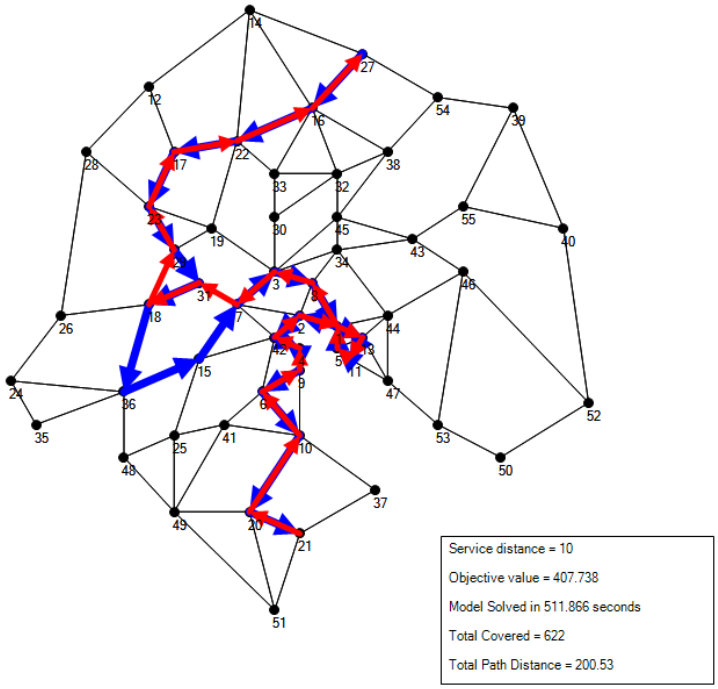


Figure 6.14 – An optimal solution to the WR-BD-MCSP problem where node *p* = 27 and node *q* = 21. In this case a maximum service distance of 10 is used, the emphasis weight on coverage = 0.6, and the emphasis on path length = 0.4. Note the use of a uni-directional loop between nodes 31, 18, 36, 15, and 7 an additional uni-directional loop between nodes 31, 18, and 29 as well as a bi-directional loop between nodes 1, 13, 11, and 5. Also note that coverage has not changed with respect to Figure 6.13; the difference is that an additional node (18) has been revisited.

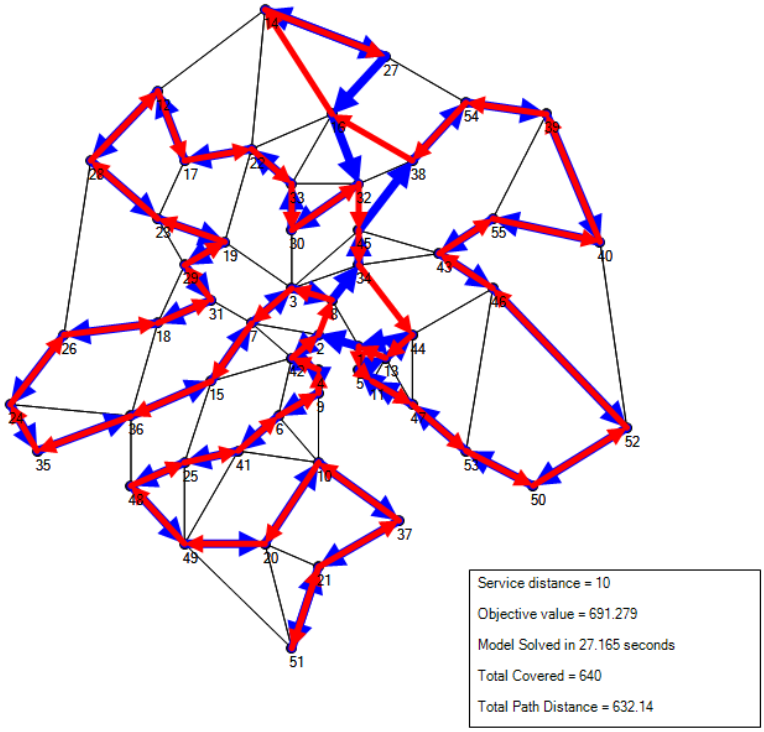


Figure 6.15 – An optimal solution to the WR-BD-MCSP problem where node *p* = 27 and node *q* = 21. In this case a maximum service distance of 10 is used, the emphasis weight on coverage = 0.99, the emphasis on path length = 0.01, and the return importance weight = 0.10. Note the 'zig-zag' crossing of the inbound and outbound paths.

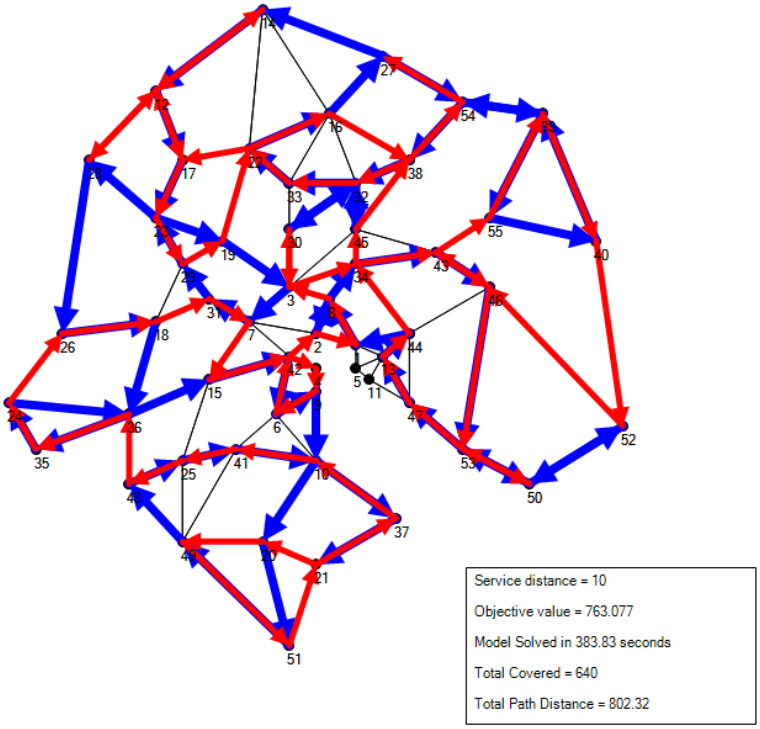


Figure 6.16 – An optimal solution to the WR-BD-MCSP problem where node *p* = 27 and node *q* = 21. In this case a maximum service distance of 10 is used, the emphasis weight on coverage = 0.99, the emphasis on path length = 0.01, and the return importance weight = 0.25. Note the 'zig-zag' crossing of the inbound and outbound paths.

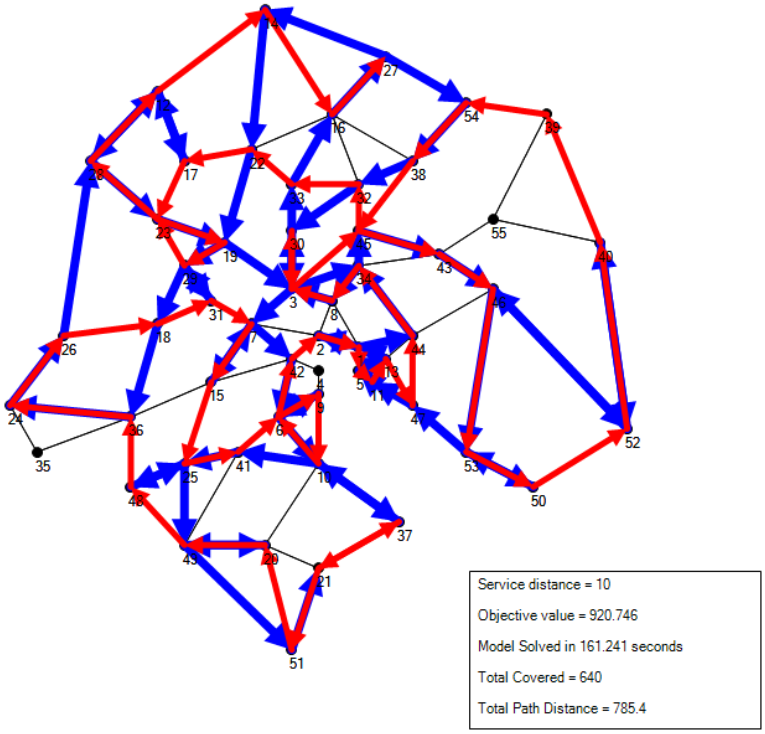


Figure 6.17 – An optimal solution to the WR-BD-MCSP problem where node p = 27 and node q = 21. In this case a maximum service distance of 10 is used, the emphasis weight on coverage = 0.99, the emphasis on path length = 0.01, and the return importance weight = 0.50. Note the 'zig-zag' crossing of the inbound and outbound paths.

Chapter 7

7.1 Concluding Remarks

This dissertation has reviewed, expanded, and developed several new models that address several critical research questions that were raised in Chapter 1. This chapter briefly reviews these research questions, the approaches taken to address them, and suggestions related to future research directions. The basis of this dissertation work stems from the fact that if one examines at system route maps for many bus transit systems in U.S. cities, an interesting pattern emerges. Routes often utilize embedded loops to increase accessibility coverage of a system at the expense of adding a marginal amount of length to the overall path. Further, such routes frequently share a common corridor with respect to traveling in opposing directions, but they may depart from each other in terms of direction. These departures in direction represent embedded loops that are traversed in only one direction. However, the literature has not explored this issue, and in fact, often discourages or outright prevents any loops from occurring. Furthermore, the literature has not accounted for travel in opposing directions, even when attempting to model transit lines. This is due in part to the roots of the covering path literature.

Chapter one notes that the MCSP problem can be formulated in a variety of ways. Virtually all covering path models utilize some form of a sub-tour elimination process that is borrowed from the Traveling Salesman Problem (TSP) literature. Due to the nature of the maximal covering shortest path problem, it is likely that a sub-tour (a cycle that is not connected to the covering path itself) will form unless constraints are added to the formulation or the formulation itself is structured in such a way so as to prevent sub-

tour formation. This is similar to the problem of sub-tours being encountered when solving the TSP problem. Accordingly, past model formulations for the MCSP have been based on model constructs found in the TSP literature that prevent the existence of sub-tours and cycles.

The seminal Shortest Covering Path (SCP) and Maximal Covering Shortest Path (MCSP) formulations of Current et. al. (1984, 1985) are built on the TSP framework developed by Dantzig, Fulkerson, and Johnson (1956). Other frameworks such as Vajda (1961), Gavish and Graves (1978), and Miller, Tucker, and Zemlin (1960) have also been used. However, all of these formulations restrict the use of a 'loop' from being used in a solution due to the fact that they prevent sub-tours and cycles from occurring. In a traveling salesman context this makes sense as one wishes to visit all nodes of a complete network exactly once. Nevertheless, as Niblett and Church (2016) demonstrated, an embedded loop could actually be present in an optimal covering path solution and thus the *a priori* exclusion of such loops could actually result in a sub-optimal solution in a maximal covering shortest path context. This dissertation presented a new, innovative 'loop agnostic' approach to the classic Current et. al. (1985) form in Chapter 3 and the alternate form of TRANSMax in Chapter 5. The issue of how loops are utilized with respect to directionality is also an interesting research question that is answered by this dissertation.

Although several loop agnostic models are developed in this dissertation to better represent the maximal covering shortest path problem, these models only capture one aspect of loop use. In a single path MCSP (i.e. a New, Revised MCSP formulated in Chapter 3 of this dissertation), a loop can be present as part of the path, as an out-and-

back path or a more complex loop which visits several other nodes before returning to a previously visited node, or as a 'lollipop' shaped route attached to the origin node or the destination node. If one assumes that the covering path can be traversed both in the outbound and inbound directions (which past work has done), any loops that are present will be utilized in both directions and is what we refer to as a bi-directional loop. When addressing the question of bi-directionality it is possible that a loop is traversed in only one direction. Such "uni-directional" loops are formed whenever inbound/outbound paths diverge and can be observed in many transit system maps, like those of Bozeman, MT; Eau Claire, WI; and San Luis Obispo, CA that were presented in this dissertation. This dissertation proposed a new route design problem called the bi-directional maximal covering shortest path problem as well as proposed two formulations for this new problem to address this real-world planning problem.

Chapter 2 of this dissertation provided an in-depth analysis of the existing covering path literature. The covering path/routing formulations that are the genesis of covering path problems are closely examined. This includes the major work from the TSP literature such as the models developed by Dantzig, Fulkerson, and Johnson (1956) as well as several other TSP frameworks such as those devised by Vajda (1961). We also present the shortest path problem and how it relates to the maximal covering shortest path problem. A review of the seminal works with respect to spatial covering models such as the Location Set Covering Problem (Toregas, et. al., 1971, 1972) and the Maximal Covering Location Problem (Church and ReVelle, 1974) are included in this chapter as well. In addition to the underlying literature this chapter also details models that have extended the MCSP formulation. This includes work dealing with issues such as

strategic route extension for transit lines (Wan and Lo, 2003; Wu and Murray, 2003; Matisziw, et. al. 2006), alternative covering path models such as TRANSMax (Curtin and Biba, 2011), and the multi-path model formulation of Boffey and Narula (1998). This chapter also introduced several algorithms and heuristics that have been used to solve path problems.

Chapter 3 of this dissertation expanded the work of Niblett and Church (2016) and introduced a new, revised form of the MCSP and Maximum Population Shortest Path (MPSP) problems. This chapter introduces a 'loop agnostic' model that allows for a loop to form as part of the covering path if it improves the value of the objective. A comparison of solutions between the revised models and the original MCSP was performed and the results were compared with respect to objective values, the use of loops, and solution times. This chapter also showed that the original MPSP model formulation introduced by Current et. al. (1985) contained explicit constraints as well as the Dantzig, Fulkerson, and Johnson TSP constraints that prevent embedded loops from occurring by preventing any node to be visited more than once. Thus, the original formulation prevents the use of loops *a priori* and precludes the determination of a truly optimal solution in certain cases. Chapter 3 also highlights the solution process that is used to add the necessary Eliminate or Attach Sub-Tour (EAST) constraints that are required to ensure a continuous, connected, covering path as well as prevent any disconnected sub-tours. These constraints allow for the embedded loops to occur within the covering path. Thus, this model neither encourages nor discourages loops to occur with a path. Because of this property, we called this model a 'loop agnostic' model as loops are used only when they yield an improvement in the objective.

Chapter 4 of this dissertation presented a heuristic that can be used in instances where computational complexity precludes one from solving an MCSP to optimality. This chapter articulated a new swapping heuristic that allows for the determination of good solutions which allow loops to form within the path when the presence of loops improves a composite objective value of distance and coverage. In particular, the heuristic process begins with the determination of the shortest path through a network. This is then set as the current best path. The heuristic then identifies the best swap that can be made by swapping a portion of the path with the insertion of a path segment that detours to another node not on the path or the insertion of an out and back path which visits a node not on the path being modified. What makes this unique is that the candidate path segment being inserted (or swapped) into the current solution can be comprised of a number of different structures including an out and back loop as well as an alternate route between two nodes on the current path. After the best insertion/substitution is made, it is repeated again until no improvement is found or until the heuristic has reached the maximum number of swap/insertion iterations. We applied this heuristic to the 55 node Swain network as well as to the Richardson, TX dataset. The heuristic was able to find optimal, or near-optimal, solutions for a majority of the problems that were tested. This is in part due to the fact that the heuristic search does not preclude the use of an embedded loop in the path/route. The heuristic also runs within a reasonable amount of computational time yielding excellent solutions in a few minutes of computer time.

In Chapter 5, a new alternate formulation of the TRANSMax model was presented. TRANSMax was developed by Curtin and Biba (2011) and represents a

variant of the Maximal Covering Shortest Path Problem. What is unique about the original TRANSMax is that it uses a different approach than that of Current et al. (1984, 1985) to eliminate sub-tours in a solution. Curtin and Biba based their model formulation on a structure proposed by Vajda (1961) to prevent sub-tours in a travelling salesman problem. One of the key issues with Curtin and Biba's formulation is that they require a path to be less than a maximum total path length and the path must utilize an exact number of arc segments. Utilizing a framework meant to solve traveling salesman problems, means that the original TRANSMax formulation prevents any embedded loops from occurring in a solution even if such a structure results in an improved objective value. This restriction was relaxed by developing a new form of Vajda's constraints, which resulted in an alternate formulation, TRANSMax II, that allows for the use of loops in a solution. The constraint that requires an exact number of arcs to be used was also relaxed. Instead, the TRANSMax II formulation is defined such that any number of arcs up to a maximum number of arcs can be used in the path. This chapter then highlights several cases where the TRANSMax II formulation determines a better objective than the original TRANSMax model. This chapter also provides a comparison between both the TRANSMax and TRANSMax II models. Computational experience and results are provided which demonstrates that the TRANSMax II model is able to outperform the original TRANSMax model while also being 'loop agnostic' in that loops can be utilized if they result in an improvement in the objective.

Chapter 6 showcased a new problem, the Bi-Directional Maximal Covering Shortest Path problem. The previous chapters of this dissertation were primarily concerned with detailing work centered around the research problem of making covering

path models loop agnostic. We show that loops have often been assumed to be inefficient structures which should never be utilized in a covering path context. Not only was this assumption shown to be incorrect, but this dissertation formulated several alternative models which are loop agnostic. That is, the new formulations allow a loop to be used, but only if it results in an improved objective. Computational results revealed that optimal solutions often utilized one or mode embedded loops, and clearly demonstrated that this long held assumption was false for realistic problems. Chapter 6 addresses an additional research question which centers on the fact that transit systems often use pathways/routes that differ spatially to some extent depending upon the direction of travel. Past covering path modeling work has been based upon the assumption that the path represents the route in both directions. When network topology prevents this (i.e. one way streets), then it is assumed that the solution can be tweaked a bit in the opposite or return direction when needed. This chapter presented examples of real world transit systems that utilize several kinds of loop structures such as a figure eight, lollipop, and barbell, where some of the route segments are traversed in only one direction creating embedded loops (called uni-directed loops). Uni-directional loops are traversed in either a clockwise or counterclockwise manner. In addition, loops are used which are bi-directional and are traversed in both a clockwise and counterclockwise manner.

Chapter 6 presented a new form of the MCSP where travel was optimized in both the inbound and outbound directions (called the Bi-Directional Maximal Covering Shortest Path problem or BD-MCSP). Two new models were formulated which allow loops to form if they improve the objective where loops may be either uni-directional or

bi-directional. The first model formulation for the BD-MCSP problem was called the Minimum Shared Arcs BD-MCSP. This formulation requires that at least a certain number of arcs must be used in common between the outbound path traveling from the origin to the destination and the inbound path traveling from the destination to the origin. We presented several optimal solutions to this problem which utilize both uni-directional and bi-directional loops. This demonstrates the flexibility afforded by accounting for bi-directional travel through the use of 'outbound' and 'inbound' paths. The chapter also suggests potential areas of future work such as multi-route bi-directional MCSP problems as well as the use of known transit data on real world networks.

Altogether, this dissertation has addressed several fundamental research issues. We have noted that the research literature has assumed that loops will not be used and that formulations should include either explicit tour-breaking constraints (Dantzig, et. al., 1954; Current et. al., 1984, 1985) or that existing formulations implicitly prevent the formation of any kind of loop/tour (Vajda, 1961; Curtin and Biba, 2011). Three new problem have been formulated – the NR-MCSP problem, the TRANSMax II problem, and the BD-MCSP problem. All new formulations are loop agnostic and allow for the possibility of a loop only if an improvement in the objective value can be made. Results in solving the NR-MCSP and TRANSMax II models both demonstrate that loops can be part of an optimal covering path solution for the classic maximal covering shortest path problem. This dissertation has also addressed the assumption that a solution to an MCSP can simply be reversed for the return direction and 'tweaked' as needed to fit the underlying network topology. Multi-path models such as Boffey and Narula (1998), transit routing models such as Wu and Murray (2003), and route extension models such

as Matisziw, et. al. (2006) are all based upon the implicit assumption that a solution can simply be reversed along a solution corridor. Even alternative MCSP formulations such as that developed by Curtin and Biba (2011) are based upon the assumption that the optimal covering path solution can simply be reversed and is essentially the same in both directions. We show this assumption to be problematic as it doesn't fully address the design elements of covering paths in use by transit agencies. Because of this shortcoming, a new problem is introduced called the Bi-Directional Maximal Covering Shortest Path (BD-MCSP) problem which optimizes travel in both directions for a route while still maintaining a loop agnostic model form.

We developed two alternate formulations of the BD-MCSP problem. The first formulation allows one to specify the minimum number of shared arcs (MSA) that should be used in common to both the outbound and inbound covering paths. We designated this formulation as the MSA-BD-MCSP. This formulation allows one to find un-supported non-dominating solutions although with a high computational cost due to the use of budget/knapsack constraints. This dissertation also included an alternative form of the BD-MCSP in which the minimum shared arc constraint is removed. In this version, called the weighted return BD-MCSP (WR-BD-MCSP) problem, we include an objective weight that allows one to specify a benefit that can be gained based upon a node being visited by both the outbound and inbound covering paths. This formulation of the BD-MCSP problem is computationally less complex than the MSA-BD-MCSP and allows one to find a set of supported non-dominating solutions within a reasonable amount of computer time. Both formulations show that loops are used in both a uni-directional and bi-directional manner. This means that both formulations are able to capture the loop

design elements found in virtually all transit systems in mid-sized cities within the United States.

Finally, it should be noted that there exist a number of questions that can be addressed in future research. One is the need to develop a heuristic for the bi-directional path/route problem. It is also desirable to consider whether other alternate forms of the MCSP problem can be developed which are loop agnostic as well. A third area of needed research is the need to expand these models to a multi-path or system framework. Finally, it would be desirable to extend these models to handle not only coverage but individual service times.

References

1. Balcik, B., and B. M. Beamon. (2008). “Facility Location in Humanitarian Relief.” *International Journal of Logistics Research and Applications* (Taylor & Francis) 11: 101-121.
2. Bellman, R. (1956). “On a routing problem.” No. RAND-P-1000 (RAND CORP): 1-7.
3. Biba, Steve, Kevin M. Curtin, and Germana Manca. (2010). "A new method for determining the population with walking access to transit." *International Journal of Geographical Information Science* 24: 347-364.
4. Boffey, Brian, and Subhash C. Narula. (1998). "Models for multi-path covering-routing problems." *Annals of Operations Research* (Springer) 82: 331-342.
5. Brotcorne, Luce, Gilbert Laporte, and Frederic Semet. (2003). "Ambulance Location and Relocation Models." *European Journal of Operational Research* (Elsevier) 147: 451-463.
6. Cathey, F. W., and D. J. Dailey. (2003). "A prescription for transit arrival/departure prediction using automatic vehicle location data." *Transportation Research Part C: Emerging Technologies* (TRB) 11: 241-264.
7. Ceder, A., and N. H. M. Wilson. (1986). “Bus network design.” *Transportation Research Part B: Methodological* (Elsevier) 20: 331-344.
8. Church, Richard L., and Charles ReVelle. (1974). "The Maximal Covering Location Problem." *Papers of the Regional Science Association* (Wiley) 32: 101-118.
9. Church, Richard L. (1974). *Synthesis of a class of public facility location problems* (Doctoral dissertation), The Johns Hopkins University, Baltimore, MD.
10. Church, Richard L., David Stoms, and Frank Davis. (1996). “Reserve Selection as a Maximal Covering Location Problem.” *Biological Conservation* (Elsevier) 76: 105-112.
11. Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4: 233-235
12. Cocking, C., E. Cevirgen, S. Helling, M. Oswald, N. Corcodel, P. Rammelsberg, G. Reinelt, and A. J. Hassel. (2009). "Colour compatibility between teeth and dental shade guides in Quinquagenarians and Septuagenarians." *Journal of oral rehabilitation* (Wiley) 36, No. 11: 848-855.

13. Current, John R., Charles ReVelle, and Jared L. Cohon. (1984). "The shortest covering path problem: an application of locational constraints to network design." *Journal of Regional Science* 24: 161-183.

14. Current, John R., Charles ReVelle, and Jared L. Cohon. (1985). "The maximum covering/shortest path problem: A multiobjective network design and routing formulation." *European Journal of Operational Research* (Elsevier) 21: 189-199.

15. Current, John R., Charles ReVelle, and Jared L. Cohon. (1988). "The minimum covering shortest path problem." *Decision Sciences* (Wiley) 19: 490-503.

16. Current, John R., Hasan Pirkul, and Erik Rolland. (1994). "Efficient Algorithms for Solving the Shortest Covering Path Problem." *Transportation Science* (INFORMS) 28: 317-327.

17. Current, John R., and David A. Schilling. (1989). "The Covering Salesman Problem." *Transportation Science* (INFORMS) 23: 208-213.

18. Current, John R., and David A. Schilling. (1994). "The median tour and maximal covering tour problems: Formulations and heuristics." *European Journal of Operational Research* (Elsevier) 73: 114-126.

19. Curtin, Kevin M., and Steve Biba. (2011). "The Transit Route Arc-Node Service Maximization problem." *European Journal of Operational Research* (Elsevier) 208: 46-56.

20. Daduna, Joachim R., and Anthony Wren. (1988). "Computer-aided transit scheduling." *Lecture Notes in Economics and Mathematical Systems* 308.

21. Dantzig, George B., R Fulkerson, and S Johnson. (1954). "Solution of a Large-Scale Traveling-Salesman Problem." *Operations Research* 2: 393-410.

22. Dantzig, George B. (1957). "Discrete-Variable Extremum Problems." *Operations Research* (INFORMS) 5: 266-277.

23. Daskin, Mark. (1983). "A Maximum Expected Covering Location Model: Formulation, Properties, and Heuristic Solution." *Transportation Science* (INFORMS) 17: 48-70.

24. Dijkstra, E W. (1959). "A Note on Two Problems in Connexion with Graphs." *Numerische Mathematik* (Springer) 1: 269-271.

25. Dorigo, M., Di Caro, G., & Gambardella, L. M. (1999). "Ant algorithms for discrete optimization." *Artificial life*, 5: 137-172.

26. Dreyfus, S. E. (1969). “An appraisal of some shortest-path algorithms.” *Operations Research*, 17: 395-412.

27. Dueck, G., & Scheuer, T. (1990). “Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing.” *Journal of computational physics*, 90: 161-175.

28. Dueck, G. (1993). “New optimization heuristics: The great deluge algorithm and the record-to-record travel.” *Journal of Computational physics*, 104: 86-92.

29. Eppstein, David. (1998). "Finding the k shortest paths." *SIAM Journal on Computing* 28: 652-673.

30. Euler, Leonhard. (1741). "Solutio Problematis ad Geometriam Situs Pertinentis." *Commentarii Academiae Scientiarum Petropolitanae* 8: 128-140.

31. Feo, T. A., & Resende, M. G. (1995). “Greedy randomized adaptive search procedures.” *Journal of Global Optimization* (Springer) 6: 109-133.

32. Figler, Scott, P. Sriraj, Eric Welch, and Nilay Yavuz. (2011). "Customer Loyalty and Chicago, Illinois, Transit Authority Buses: Results from 2008 Customer Satisfaction Survey." *Transportation Research Record: Journal of the Transportation Research Board* (TRB) 2216: 148-156.

33. Furth, Peter G., and F. Brian Day. (1985). "Transit routing and scheduling strategies for heavy-demand corridors (Abridgment)." *Transportation Research Record* (TRB) 1011.

34. Gavish, Bezalel. (1983). “Formulations and Algorithms for the Capacitated Minimal Directed Tree Problem.” *Journal of the Association for Computing Machinery* (ACM) 30: 118-132.

35. Gavish, Bezalel, and Stephen C. Graves. (1978). "The travelling salesman problem and related problems." (Working Paper) *Operations Research Center* (MIT).

36. Gendreau, M., Hertz, A., Laporte, G., & Stan, M. (1998). A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research*, 46: 330-335.

37. Gleason, John M. (1975). "A set covering approach to bus stop location." *Omega* (Elsevier) 3: 605-608.

38. Golden, B. L., & Skiscim, C. C. (1986). “Using simulated annealing to solve routing and location problems.” *Naval Research Logistics Quarterly*, 33: 261-279.

39. Gomory, Ralph. (1958). "Outline of an Algorithm for Integer Solutions to Linear Programs," *Bulletin of the American Mathematical Society*, 64: 275-278.

40. Gomory, Ralph. (1960). *An algorithm for the mixed integer problem*. No. RAND-P-1885. RAND Corporation, Santa Monica, CA.

41. Hakimi, S. L. (1964). “Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph.” *Operations Research* (INFORMS) 12: 450-459.

42. Hart, J. P., & Shogan, A. W. (1987). “Semi-greedy heuristics: An empirical study.” *Operations Research Letters*, 6: 107-114.

43. Hogan, Kathleen, and Charles ReVelle. (1986). “Concepts and Applications of Backup Coverage.” *Management Science* (INFORMS) 32: 1434-1444.

44. Hosage, C. M., & Goodchild, M. F. (1986). Discrete space location-allocation solutions from genetic algorithms. *Annals of Operations Research*, 6: 35-46.

45. Hsiao, Shirley, Jian Lu, James Sterling, and Matthew Weatherford. (1997). "Use of geographic information system for analysis of transit pedestrian access." *Transportation Research Record: Journal of the Transportation Research Board* (TRB), 1604: 50-59.

46. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). “Optimization by simulated annealing.” *Science*, 220: 671-680.

47. Lawler, Eugene L., Jan Karel Lenstra, AHG Rinnooy Kan, and David B. Shmoys. (1985). *The traveling salesman problem*. John Wiley and Sons.

48. Levinson, Herbert S., and Orikaye Brown-West. (1984). *Estimating bus ridership*. No. 994.

49. Levinson, H.S. (1992). “System and service planning.” In: Gray, L.A., Hoel, L.A. (Eds.), *Public Transportation, Second ed.* (Prentice-Hall) Englewood Cliffs, NJ: 369–406.

50. Li, Meng, Yafeng Yin, Wei-Bin Zhang, Kun Zhou, and Hideki Nakamura. (2011). "Modeling and implementation of adaptive transit signal priority on actuated control systems." *Computer‐Aided Civil and Infrastructure Engineering* (Wiley) 26: 270-284.

51. Lin, S. (1965). Computer solutions of the traveling salesman problem. *The Bell System Technical Journal, 44*(10), 2245-2269.

52. Maranzana, F. E. (1964). “On the Location of Supply Points to Minimize Transport Costs.” *Operations Research Quarterly* (Pergamon Press) 15: 261-270.

53. Marianov, Vladimir, and Charles ReVelle. (1996). “The Queuing Maximal availability location problem: A model for the siting of emergency vehicles.” *European Journal of Operational Research* (Elsevier) 93: 110-120.

54. Martins, E. D. Q. V. (1984). “An algorithm for ranking paths that may contain cycles.” *European Journal of Operational Research* (Elsevier), 18: 123-130.

55. Matisziw, Timothy C., Alan T. Murray, and Changjoo Kim. (2006). "Strategic route extension in transit networks." *European journal of operational research* (Elsevier) 171: 661-673.

56. Miller, Clair E., Albert W. Tucker, and Richard A. Zemlin. (1960). "Integer programming formulation of traveling salesman problems." *Journal of the Association for Computing Machinery* (ACM) 7: 326-329.

57. Mladenović, N., & Hansen, P. (1997). “Variable neighborhood search.” *Computers & Operations Research,* 24: 1097-1100.

58. Murray, Alan T., R. Davis, R. J. Stimson, and L. Ferreira. (1998). “Public transportation access.” *Transportation Research Part D: Transport and Environment* (Elsevier) 3: 319-328.

59. Murray, Alan T. (2001). “Strategic Analysis of public transport coverage.” *Socio-Economic Planning Sciences* (Elsevier) 35: 175-188.

60. Murray, Alan T. and R. Davis. (2001). “Equity in regional service provision.” *Journal of Regional Science* (Wiley) 41: 557-600

61. Murray, Alan T., and X Wu. (2003). “Accessibility tradeoffs in public transit planning.” *Journal of Geographical Systems* (Springer) 5: 93-107

62. Murray, Alan T. (2003). “A coverage model for improving public transit system accessibility and expanding access.” *Annals of Operations Research* (Springer) 123: 143-156.

63. Narula, S. C., Ogbu, U. I., & Samuelsson, H. M. (1977). Technical Note—An Algorithm for the p-Median Problem. *Operations Research*, 25: 709-713.

64. Newman, Peter, and Jeffrey Kenworthy. (1999). *Sustainability and cities: overcoming automobile dependence*. Island Press.

65. Niblett, Timothy J., and Richard L. Church. (2016). "The Shortest Covering Path Problem: A New Perspective and Model." *International Regional Science Review* (Sage Publications) 39: 1-21.

66. Norden, John. (1625). *England, an Intended Guyde for English Travailers.* London: Edward Allde.

67. O'Neill, Wende A., R. Douglas Ramsey, and JaChing Chou. (1992). "Analysis of transit service areas using geographic information systems." *Transportation Research Record* (TRB) 1364.

68. Orden, Alex. (1956). "The Transhipment Problem." *Management Science* (INFORMS) 2: 276-285.

69. Orman, A. J., and H. P. Williams. (2006). "A Survey of Different Integer Programming Formulations of the Traveling Salesman Problem" in Kontoghiorhes, E. J., and Cristian Gatu, eds. *Optimisation, Economic and Financial Analysis* (Springer): 93-106.

70. ReVelle, Charles, and Kathleen Hogan. (1989). "The Maximum Availability Location Problem." *Transportation Science* (INFORMS) 23: 192-200.

71. Roberts, Dar, M. Gardner, R. L. Church, S. Ustin, G. Scheer, R. Green. (1998). "Mapping Chaparral in the Santa Monica Mountains Using Multiple Endmember Spectral Mixture Models." *Remote Sensing of Environment* (Elsevier) 65: 267-279.

72. Rolland, E., Schilling, D. A., & Current, J. R. (1997). "An efficient tabu search procedure for the p-median problem." *European Journal of Operational Research* (Elsevier) 96: 329-342.

73. Rousseau, J. M. (1985). *Computer Scheduling of Public Transport 2*. Elsevier, New York, NY.

74. Schilling, David, D. Jack Elzinga, Jared Cohon, Richard Church, and Charles ReVelle. (1979). "The Team/Fleet Models for Simultaneous Facility and Equipment Siting." *Transportation Science* (INFORMS) 13: 163-175.

75. Schrijver, Alexander. (2003). *Combinatorial Optimization: Polyhedra and Efficiency.* New York: Springer-Verlag.

76. Schrijver, Alexander. (2012). "On the history of the shortest path problem." *Documenta Mathematica*: 155-167.

77. Shier, D. R. (1979). “On algorithms for finding the k shortest paths in a network.” *Networks*, 9: 195-214.

78. Teitz, Michael B., and Polly Bart. (1968). "Heuristic methods for estimating the generalized vertex median of a weighted graph." *Operations Research* (INFORMS) 16: 955-961.

79. Toregas, Constantine, Ralph Swain, Charles ReVelle, and Lawrence Bergman. (1971). "The Location of Emergency Service Facilities." *Operations Research* (INFORMS) 19: 1363-1373.

80. Toregas, Constantine, and Charles ReVelle. (1972). “Optimal Location Under Time or Distance Constraints.” *Papers in Regional Science* (Wiley) 28: 133-144.

81. Tyrinopoulos, Yannis, and Constantinos Antoniou. (2008). "Public transit user satisfaction: Variability and policy implications." *Transport Policy* (Elsevier) 15: 260-272.

82. Vajda, Steven. (1961). *Mathematical Programming*. Addison-Wesley: Reading, MA.

83. Wan, Quentin K., and Hong K. Lo. (2003). “A Mixed Integer Formulation for Multiple-Route Transit Network Design.” *Journal of Mathematical Modelling and Algorithms* (Kluwer) 2: 299-308.

84. Weinstein, Aaron. (2000). "Customer satisfaction among transit riders: how customers rank the relative importance of various service attributes." *Transportation Research Record: Journal of the Transportation Research Board* (TRB) 1735: 123-132.

85. Wren, Anthony, and David O. Wren. (1995). "A genetic algorithm for public transport driver scheduling." *Computers & Operations Research* (Elsevier) 22: 101-110.

86. Wu, Changshan, and Alan T. Murray. (2005). "Optimizing public transit quality and system access: the multiple-route, maximal covering/shortest-path problem." *Environment and Planning B: Planning and Design* (Pion) 32: 163-178.

87. Yen, J. Y. (1971). “Finding the *k* shortest loopless paths in a network.” *Management Science,* 17: 712-716.

88. Yu, Bin, William HK Lam, and Mei Lam Tam. (2011). "Bus arrival time prediction at bus stop with multiple routes." *Transportation Research Part C: Emerging Technologies* (TRB) 19:1157-1170.

Appendix I

Table 1 - Results for solving the NR-MCSP applied on the hypothetical Swain network. The origin is node 27 and the destination is node 21.

Cover Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective	Time (Seconds)
0	1	0	171	51.92	-51.920	0.012
0.01	0.99	0	206	52.06	-49.479	0.012
0.02	0.98	0	206	52.06	-46.899	0.012
0.03	0.97	0	206	52.06	-44.318	0.013
0.04	0.96	0	260	54	-41.440	0.014
0.05	0.95	0	260	54	-38.300	0.016
0.06	0.94	0	331	58.47	-35.102	0.195
0.10	0.9	0	331	58.47	-19.523	1.257
⋮	Same Solution Values					⋮
0.11	0.89	0	366	62.47	-15.338	2.027
0.12	0.88	0	417	68.94	-10.627	2.422
⋮	Same Solution Values					⋮
0.24	0.76	0	417	68.94	47.686	5.063
0.25	0.75	0	437	75.45	52.663	5.461
⋮	Same Solution Values					⋮
0.28	0.72	0	437	75.45	68.036	5.457
0.29	0.71	0	464	86.32	73.273	5.267
0.33	0.67	0	464	86.32	95.286	4.264
0.34	0.66	0	473	90.83	100.872	3.657
⋮	Same Solution Values					⋮
0.38	0.62	0	473	90.83	123.425	4.410
0.39	0.61	0	483	97.08	129.151	4.624
⋮	Same Solution Values					⋮
0.42	0.58	0	483	97.08	146.554	7.202
0.43	0.57	0	504	112.9	152.367	18.256
0.44	0.56	0	513	119.83	158.615	18.879
0.45	0.55	0	544	144.34	165.413	10.447
⋮	Same Solution Values					⋮
0.49	0.51	0	544	144.34	192.947	7.746
0.5	0.5	0	556	156.15	199.925	14.119
0.51	0.49	0	584	185.08	207.151	32.268
0.52	0.48	0	584	185.08	214.842	28.135

Cover Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective	Time (Seconds)
0.53	0.47	0	584	185.08	222.532	37.052
0.54	0.46	0	596	198.81	230.387	36.186
0.55	0.45	0	596	198.81	238.336	25.409
0.56	0.44	0	596	198.81	246.284	24.653
0.57	0.43	0	599	202.68	254.278	16.336
0.58	0.42	0	603	208.04	262.363	12.119
⋮	Same Solution Values					⋮
0.61	0.39	0	603	208.04	286.694	10.162
0.62	0.38	0	610	219.12	294.934	7.903
⋮	Same Solution Values					⋮
0.66	0.34	0	610	219.12	328.099	0.632
0.67	0.33	0	613	224.96	336.473	0.726
⋮	Same Solution Values					⋮
0.7	0.3	0	613	224.96	361.612	0.151
0.71	0.29	0	618	236.73	370.128	0.121
0.72	0.28	0	618	236.73	378.676	0.287
0.73	0.27	0	618	236.73	387.223	3.694
0.74	0.26	0	630	269.82	396.047	9.788
⋮	Same Solution Values					⋮
0.79	0.21	0	630	269.82	441.038	9.543
0.8	0.2	0	637	297.55	450.090	13.271
0.81	0.19	0	637	297.55	459.436	8.724
0.82	0.18	0	637	297.55	468.781	8.258
0.83	0.17	0	640	312.11	478.141	10.997
⋮	Same Solution Values					⋮
0.99	0.01	0	640	312.11	630.479	11.991
0	1	2.5	210	51.92	-51.920	0.010
⋮	Same Solution Values					⋮
0.03	0.97	2.5	210	51.92	-44.062	0.017
0.04	0.96	2.5	384	58.33	-40.637	0.034
⋮	Same Solution Values					⋮
0.13	0.87	2.5	384	58.33	-0.827	0.441
0.14	0.86	2.5	401	61.07	3.620	0.507
⋮	Same Solution Values					⋮
0.2	0.8	2.5	401	61.07	31.344	1.557
0.21	0.79	2.5	417	65.07	36.165	1.714

Cover Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective	Time (Seconds)
⋮	Same Solution Values					⋮
0.24	0.76	2.5	417	65.07	50.627	3.733
0.25	0.75	2.5	437	71.58	55.565	3.379
⋮	Same Solution Values					⋮
0.28	0.72	2.5	437	71.58	70.822	3.725
0.29	0.71	2.5	464	82.45	76.021	4.062
⋮	Same Solution Values					⋮
0.33	0.67	2.5	464	82.45	97.879	2.482
0.34	0.66	2.5	473	86.96	103.426	2.643
⋮	Same Solution Values					⋮
0.38	0.62	2.5	473	86.96	125.825	3.548
0.39	0.61	2.5	483	93.21	131.512	3.635
⋮	Same Solution Values					⋮
0.42	0.58	2.5	483	93.21	148.798	5.962
0.43	0.57	2.5	504	109.03	154.573	10.241
0.44	0.56	2.5	528	127.62	160.853	15.572
0.45	0.55	2.5	552	147.02	167.539	15.382
⋮	Same Solution Values					⋮
0.49	0.51	2.5	552	147.02	195.500	9.594
0.5	0.5	2.5	556	151.02	202.490	15.018
0.51	0.49	2.5	556	151.02	209.560	23.493
0.52	0.48	2.5	577	173.52	216.750	45.418
0.53	0.47	2.5	577	173.52	224.256	67.049
0.54	0.46	2.5	596	195.4	231.956	56.156
0.55	0.45	2.5	596	195.4	239.870	38.809
0.56	0.44	2.5	596	195.4	247.784	27.197
0.57	0.43	2.5	599	199.27	255.744	19.715
0.58	0.42	2.5	603	204.63	263.795	16.830
⋮	Same Solution Values					⋮
0.61	0.39	2.5	603	204.63	288.024	11.709
0.62	0.38	2.5	610	215.71	296.230	6.273
⋮	Same Solution Values					⋮
0.7	0.3	2.5	610	215.71	362.287	0.943
0.71	0.29	2.5	615	227.48	370.681	1.201
0.72	0.28	2.5	618	235.13	379.124	1.383
0.73	0.27	2.5	618	235.13	387.655	13.055

Cover Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective	Time (Seconds)
0.74	0.26	2.5	630	268.22	396.463	16.288
⋮	Same Solution Values					⋮
0.77	0.23	2.5	630	268.22	423.409	11.829
0.78	0.22	2.5	637	292.78	432.448	11.840
⋮	Same Solution Values					⋮
0.82	0.18	2.5	637	292.78	469.640	3.342
0.83	0.17	2.5	640	307.34	478.952	2.914
⋮	Same Solution Values					⋮
0.99	0.01	2.5	640	307.34	630.527	3.089
0	1	5	387	51.92	-51.920	0.012
⋮	Same Solution Values					⋮
0.06	0.94	5	387	51.92	-25.585	0.024
0.07	0.93	5	456	56.39	-20.523	0.032
⋮	Same Solution Values					⋮
0.21	0.79	5	456	56.39	51.212	0.306
0.22	0.78	5	463	58.33	56.363	0.442
⋮	Same Solution Values					⋮
0.28	0.72	5	463	58.33	87.642	6.701
0.29	0.71	5	507	75.71	93.276	7.669
⋮	Same Solution Values					⋮
0.33	0.67	5	507	75.71	116.584	10.039
0.34	0.66	5	512	78.2	122.468	12.577
⋮	Same Solution Values					⋮
0.37	0.63	5	512	78.2	140.174	20.992
0.38	0.62	5	535	91.91	146.316	26.828
⋮	Same Solution Values					⋮
0.42	0.58	5	535	91.91	171.392	50.120
0.43	0.57	5	568	116.59	177.784	58.765
⋮	Same Solution Values					⋮
0.5	0.5	5	568	116.59	225.705	190.784
0.51	0.49	5	599	148.37	232.789	277.290
0.52	0.48	5	611	160.88	240.498	153.295
⋮	Same Solution Values					⋮
0.63	0.37	5	611	160.88	325.404	20.529
0.64	0.36	5	617	171.4	333.176	21.174

Cover Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective	Time (Seconds)
⋮	Same Solution Values					⋮
0.76	0.24	5	617	171.4	427.784	120.371
0.77	0.23	5	633	223.39	436.030	213.333
0.78	0.22	5	633	223.39	444.594	124.859
0.79	0.21	5	633	223.39	453.158	93.806
0.8	0.2	5	640	251.01	461.798	72.717
⋮	Same Solution Values					⋮
0.99	0.01	5	640	251.01	631.090	1.937
0	1	7.5	499	51.92	-51.920	0.011
⋮	Same Solution Values					⋮
0.17	0.83	7.5	499	51.92	41.736	0.034
0.18	0.82	7.5	534	59.5	47.330	0.089
⋮	Same Solution Values					⋮
0.22	0.78	7.5	534	59.5	71.070	0.102
0.23	0.77	7.5	556	65.77	77.237	0.081
⋮	Same Solution Values					⋮
0.3	0.7	7.5	556	65.77	120.761	0.108
0.31	0.69	7.5	565	69.8	126.988	0.197
⋮	Same Solution Values					⋮
0.35	0.65	7.5	565	69.8	152.380	0.740
0.36	0.64	7.5	576	75.92	158.771	0.698
⋮	Same Solution Values					⋮
0.51	0.49	7.5	576	75.92	256.559	20.727
0.52	0.48	7.5	613	115.05	263.536	21.613
⋮	Same Solution Values					⋮
0.57	0.43	7.5	613	115.05	299.939	27.409
0.58	0.42	7.5	623	128.6	307.328	31.688
⋮	Same Solution Values					⋮
0.77	0.23	7.5	623	128.6	450.132	745.392
0.78	0.22	7.5	627	142.03	457.813	1324.970
0.79	0.21	7.5	634	168.27	465.523	4097.080
0.8	0.2	7.5	634	168.27	473.546	4018.380
0.81	0.19	7.5	640	192.79	481.770	2816.100
⋮	Same Solution Values					⋮
0.99	0.01	7.5	640	192.79	631.672	85.813

Cover Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective	Time (Seconds)
0	1	10	537	51.92	-51.920	0.012
⋮	Same Solution Values					⋮
0.07	0.93	10	537	51.92	-10.696	0.022
0.08	0.92	10	552	53.2	-4.784	0.021
⋮	Same Solution Values					⋮
0.24	0.76	10	552	53.2	92.048	0.191
0.25	0.75	10	571	59.5	98.125	0.764
0.26	0.74	10	571	59.5	104.430	2.921
0.27	0.73	10	592	67.11	110.850	3.324
⋮	Same Solution Values					⋮
0.38	0.62	10	592	67.11	183.352	82.724
0.39	0.61	10	600	72.02	190.068	104.171
0.4	0.6	10	613	80.38	196.972	121.146
⋮	Same Solution Values					⋮
0.44	0.56	10	613	80.38	224.707	196.901
0.45	0.55	10	620	85.94	231.733	192.819
⋮	Same Solution Values					⋮
0.64	0.36	10	620	85.94	365.862	1954.260
0.65	0.35	10	633	109.45	373.143	1461.560
⋮	Same Solution Values					⋮
0.82	0.18	10	633	109.45	499.359	17433.200
0.83	0.17	10	638	133.77	506.799	92232.600
0.84	0.16	10	638	133.77	514.517	102985.000
0.85	0.15	10	638	133.77	522.235	140982.000
0.86	0.14	10	640	145.43	530.040	113137.000
⋮	Same Solution Values					⋮
0.99	0.01	10	640	145.43	632.146	16544.900
0	1	12.5	571	51.92	-51.920	0.013
⋮	Same Solution Values					⋮
0.22	0.78	12.5	571	51.92	85.122	0.075
0.23	0.77	12.5	595	59	91.420	0.169
⋮	Same Solution Values					⋮
0.32	0.68	12.5	595	59	150.280	5.550
0.33	0.67	12.5	613	67.54	157.038	6.847
⋮	Same Solution Values					⋮

Cover Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective	Time (Seconds)
0.47	0.53	12.5	613	67.54	252.314	144.147
0.48	0.52	12.5	617	71.16	259.157	180.963
0.49	0.51	12.5	617	71.16	266.038	230.913
0.5	0.5	12.5	617	71.16	272.920	301.393
0.51	0.49	12.5	629	83.32	279.963	297.954
⋮	Same Solution Values					⋮
0.72	0.28	12.5	629	83.32	429.550	108.877
0.73	0.27	12.5	633	93.71	436.788	131.760
⋮	Same Solution Values					⋮
0.79	0.21	12.5	633	93.71	480.391	1691.100
0.8	0.2	12.5	638	113.41	487.718	2734.620
⋮	Same Solution Values					⋮
0.88	0.12	12.5	638	113.41	547.831	2587.130
0.89	0.11	12.5	640	128.26	555.491	1322.350
⋮	Same Solution Values					⋮
0.99	0.01	12.5	640	128.26	632.317	45.647
0	1	15	588	51.92	-51.920	0.012
⋮	Same Solution Values					⋮
0.26	0.74	15	588	51.92	114.459	0.071
0.27	0.73	15	609	59.5	120.995	0.089
⋮	Same Solution Values					⋮
0.38	0.62	15	609	59.5	194.530	7.362
0.39	0.61	15	614	62.67	201.231	9.354
⋮	Same Solution Values					⋮
0.43	0.57	15	614	62.67	228.298	30.931
0.44	0.56	15	622	68.92	235.085	37.394
⋮	Same Solution Values					⋮
0.52	0.48	15	622	68.92	290.358	177.047
0.53	0.47	15	629	76.53	297.401	184.540
⋮	Same Solution Values					⋮
0.72	0.28	15	629	76.53	431.452	1596.750
0.73	0.27	15	633	87.24	438.535	1818.250
⋮	Same Solution Values					⋮
0.79	0.21	15	633	87.24	481.750	12505.600
0.8	0.2	15	640	114.9	489.020	25344.300

Cover Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective	Time (Seconds)
⋮	Same Solution Values					⋮
0.99	0.01	15	640	114.9	632.451	1263.050
0	1	17.5	590	51.92	-51.920	0.014
⋮	Same Solution Values					⋮
0.06	0.94	17.5	590	51.92	-13.405	0.030
0.07	0.93	17.5	610	53.2	-6.776	0.028
⋮	Same Solution Values					⋮
0.22	0.78	17.5	610	53.2	92.704	0.040
0.23	0.77	17.5	623	56.92	99.462	0.036
⋮	Same Solution Values					⋮
0.48	0.52	17.5	623	56.92	269.442	0.147
0.49	0.51	17.5	629	62.56	276.304	0.114
⋮	Same Solution Values					⋮
0.71	0.29	17.5	629	62.56	428.448	47.197
0.72	0.28	17.5	640	90.78	435.382	59.926
⋮	Same Solution Values					⋮
0.99	0.01	17.5	640	90.78	632.692	14.595
0	1	20	616	51.92	-51.920	0.013
⋮	Same Solution Values					⋮
0.25	0.75	20	616	51.92	115.060	0.040
0.26	0.74	20	631	56.92	121.939	0.050
⋮	Same Solution Values					⋮
0.29	0.71	20	631	56.92	142.577	0.047
0.3	0.7	20	635	58.61	149.473	0.045
⋮	Same Solution Values					⋮
0.72	0.28	20	635	58.61	440.789	1.259
0.73	0.27	20	638	66.62	447.753	1.848
⋮	Same Solution Values					⋮
0.85	0.15	20	638	66.62	532.307	4.185
0.86	0.14	20	640	78.2	539.452	4.421
⋮	Same Solution Values					⋮
0.99	0.01	20	640	78.2	632.818	2.918
0	1	22.5	638	51.92	-51.920	0.014
⋮	Same Solution Values					⋮
0.82	0.18	22.5	638	51.92	513.814	0.041

Cover Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective	Time (Seconds)
0.83	0.17	22.5	640	61.32	520.776	0.045
⋮	Same Solution Values					⋮
0.99	0.01	22.5	640	61.32	632.987	0.725
0	1	25	638	51.92	-51.920	0.014
⋮	Same Solution Values					⋮
0.79	0.21	25	638	51.92	493.117	0.932
0.8	0.2	25	640	59.65	500.070	1.104
⋮	Same Solution Values					⋮
0.99	0.01	25	640	59.65	633.004	0.910
0	1	27.5	640	51.92	-51.920	0.015
⋮	Same Solution Values					⋮
0.99	0.01	27.5	640	51.92	633.081	0.035
0	1	30	640	51.92	-51.920	0.016
⋮	Same Solution Values					⋮
0.99	0.01	30	640	51.92	633.081	0.035
0	1	32.5	640	51.92	-51.920	0.016
⋮	Same Solution Values					⋮
0.99	0.01	32.5	640	51.92	633.081	0.035
0	1	35	640	51.92	-51.920	0.016
⋮	Same Solution Values					⋮
0.99	0.01	35	640	51.92	633.081	0.035
0	1	50	640	51.92	-51.920	0.016
⋮	Same Solution Values					⋮
0.99	0.01	50	640	51.92	633.081	0.023

Table 2 – Results for solving the MCSP applied on the hypothetical Swain network

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
0	1	0	171	51.92	-51.920
0.01	0.99	0	206	52.06	-49.479
0.02	0.98	0	206	52.06	-46.899
0.03	0.97	0	206	52.06	-44.318
0.04	0.96	0	260	54	-41.440
0.05	0.95	0	260	54	-38.300
0.06	0.94	0	331	58.47	-35.102
⋮	Same Solution Values				⋮
0.11	0.89	0	331	58.47	-15.628
0.12	0.88	0	352	61.07	-11.502
⋮	Same Solution Values				⋮
0.16	0.84	0	352	61.07	5.021
0.17	0.83	0	443	78.71	9.981
⋮	Same Solution Values				⋮
0.32	0.68	0	443	78.71	88.237
0.33	0.67	0	473	93.25	93.613
0.34	0.66	0	482	97.76	99.358
⋮	Same Solution Values				⋮
0.38	0.62	0	482	97.76	122.549
0.39	0.61	0	492	104.01	128.434
⋮	Same Solution Values				⋮
0.42	0.58	0	492	104.01	146.314
0.43	0.57	0	513	119.83	152.287
0.44	0.56	0	513	119.83	158.615
0.45	0.55	0	544	144.34	165.413
⋮	Same Solution Values				⋮

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
0.49	0.51	0	544	144.34	192.947
0.5	0.5	0	556	156.15	199.925
0.51	0.49	0	584	185.08	207.151
0.52	0.48	0	584	185.08	214.842
0.53	0.47	0	584	185.08	222.532
0.54	0.46	0	596	198.81	230.387
0.55	0.45	0	596	198.81	238.336
0.56	0.44	0	596	198.81	246.284
0.57	0.43	0	599	202.68	254.278
0.58	0.42	0	603	208.04	262.363
⋮	Same Solution Values				⋮
0.61	0.39	0	603	208.04	286.694
0.62	0.38	0	610	219.12	294.934
⋮	Same Solution Values				⋮
0.66	0.34	0	610	219.12	328.099
0.67	0.33	0	613	224.96	336.473
⋮	Same Solution Values				⋮
0.7	0.3	0	613	224.96	361.612
0.71	0.29	0	618	236.73	370.128
0.72	0.28	0	618	236.73	378.676
0.73	0.27	0	618	236.73	387.223
0.74	0.26	0	630	270.19	395.951
⋮	Same Solution Values				⋮
0.79	0.21	0	630	270.19	440.960
0.8	0.2	0	637	297.55	450.090
⋮	Same Solution Values				⋮
0.83	0.17	0	637	297.55	478.127

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
0.84	0.16	0	640	312.8	487.552
⋮	Same Solution Values				⋮
0.99	0.01	0	640	312.8	630.472
0	1	2.5	210	51.92	-51.920
⋮	Same Solution Values				⋮
0.03	0.97	2.5	210	51.92	-44.062
0.04	0.96	2.5	384	58.33	-40.637
⋮	Same Solution Values				⋮
0.13	0.87	2.5	384	58.33	-0.827
0.14	0.86	2.5	401	61.07	3.620
⋮	Same Solution Values				⋮
0.24	0.76	2.5	401	61.07	49.827
0.25	0.75	2.5	421	67.58	54.565
0.26	0.74	2.5	443	75.3	59.458
⋮	Same Solution Values				⋮
0.32	0.68	2.5	443	75.3	90.556
0.33	0.67	2.5	473	89.84	95.897
0.34	0.66	2.5	482	94.35	101.609
⋮	Same Solution Values				⋮
0.38	0.62	2.5	482	94.35	124.663
0.39	0.61	2.5	492	100.6	130.514
⋮	Same Solution Values				⋮
0.42	0.58	2.5	492	100.6	148.292
0.43	0.57	2.5	513	116.42	154.231
0.44	0.56	2.5	537	135.15	160.596
0.45	0.55	2.5	552	147.02	167.539
⋮	Same Solution Values				⋮

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
0.51	0.49	2.5	552	147.02	209.480
0.52	0.48	2.5	581	178.32	216.526
0.53	0.47	2.5	584	181.67	224.135
0.54	0.46	2.5	596	195.4	231.956
0.55	0.45	2.5	596	195.4	239.870
0.56	0.44	2.5	596	195.4	247.784
0.57	0.43	2.5	599	199.27	255.744
0.58	0.42	2.5	603	204.63	263.795
⋮	Same Solution Values				⋮
0.61	0.39	2.5	603	204.63	288.024
0.62	0.38	2.5	610	215.71	296.230
⋮	Same Solution Values				⋮
0.7	0.3	2.5	610	215.71	362.287
0.71	0.29	2.5	615	227.48	370.681
0.72	0.28	2.5	618	235.13	379.124
0.73	0.27	2.5	618	235.13	387.655
0.74	0.26	2.5	630	268.59	396.367
⋮	Same Solution Values				⋮
0.77	0.23	2.5	630	268.59	423.324
0.78	0.22	2.5	637	292.78	432.448
⋮	Same Solution Values				⋮
0.83	0.17	2.5	637	292.78	478.937
0.84	0.16	2.5	640	308.03	488.315
⋮	Same Solution Values				⋮
0.99	0.01	2.5	640	308.03	630.520
0	1	5	387	51.92	-51.920
⋮	Same Solution Values				⋮

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
0.06	0.94	5	387	51.92	-25.585
0.07	0.93	5	456	56.39	-20.523
⋮	Same Solution Values				⋮
0.21	0.79	5	456	56.39	51.212
0.22	0.78	5	463	58.33	56.363
⋮	Same Solution Values				⋮
0.28	0.72	5	463	58.33	87.642
0.29	0.71	5	507	75.71	93.276
⋮	Same Solution Values				⋮
0.33	0.67	5	507	75.71	116.584
0.34	0.66	5	512	78.2	122.468
⋮	Same Solution Values				⋮
0.37	0.63	5	512	78.2	140.174
0.38	0.62	5	535	91.91	146.316
⋮	Same Solution Values				⋮
0.42	0.58	5	535	91.91	171.392
0.43	0.57	5	568	116.59	177.784
⋮	Same Solution Values				⋮
0.5	0.5	5	568	116.59	225.705
0.51	0.49	5	599	148.37	232.789
0.52	0.48	5	611	160.88	240.498
⋮	Same Solution Values				⋮
0.63	0.37	5	611	160.88	325.404
0.64	0.36	5	617	171.4	333.176
⋮	Same Solution Values				⋮
0.76	0.24	5	617	171.4	427.784
0.77	0.23	5	627	203.81	435.914

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
0.78	0.22	5	627	203.81	444.222
0.79	0.21	5	630	214.67	452.619
0.8	0.2	5	637	242.31	461.138
⋮	Same Solution Values				⋮
0.83	0.17	5	637	242.31	487.517
0.84	0.16	5	640	257.56	496.390
⋮	Same Solution Values				⋮
0.99	0.01	5	640	257.56	631.024
0	1	7.5	497	51.92	-51.920
⋮	Same Solution Values				⋮
0.17	0.83	7.5	497	51.92	41.396
0.18	0.82	7.5	532	59.5	46.970
⋮	Same Solution Values				⋮
0.22	0.78	7.5	532	59.5	70.630
0.23	0.77	7.5	554	65.77	76.777
⋮	Same Solution Values				⋮
0.3	0.7	7.5	554	65.77	120.161
0.31	0.69	7.5	563	69.8	126.368
⋮	Same Solution Values				⋮
0.35	0.65	7.5	563	69.8	151.680
0.36	0.64	7.5	574	75.92	158.051
⋮	Same Solution Values				⋮
0.51	0.49	7.5	574	75.92	255.539
0.52	0.48	7.5	611	115.05	262.496
⋮	Same Solution Values				⋮
0.57	0.43	7.5	611	115.05	298.799
0.58	0.42	7.5	621	128.6	306.168

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
⋮	Same Solution Values				⋮
0.7	0.3	7.5	621	128.6	396.120
0.71	0.29	7.5	627	143.02	403.694
⋮	Same Solution Values				⋮
0.78	0.22	7.5	627	143.02	457.596
0.79	0.21	7.5	634	169.26	465.315
0.8	0.2	7.5	640	192.79	473.442
⋮	Same Solution Values				⋮
0.99	0.01	7.5	640	192.79	631.672
0	1	10	523	51.92	-51.920
⋮	Same Solution Values				⋮
0.07	0.93	10	523	51.92	-11.676
0.08	0.92	10	538	53.2	-5.904
⋮	Same Solution Values				⋮
0.24	0.76	10	538	53.2	88.688
0.25	0.75	10	557	59.5	94.625
0.26	0.74	10	557	59.5	100.790
0.27	0.73	10	578	67.11	107.070
⋮	Same Solution Values				⋮
0.38	0.62	10	578	67.11	178.032
0.39	0.61	10	586	72.02	184.608
0.4	0.6	10	599	80.38	191.372
⋮	Same Solution Values				⋮
0.44	0.56	10	599	80.38	218.547
0.45	0.55	10	618	95.71	225.460
⋮	Same Solution Values				⋮
0.69	0.31	10	618	95.71	396.750

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
0.7	0.3	10	633	130.02	404.094
⋮	Same Solution Values				⋮
0.81	0.19	10	633	130.02	488.026
0.82	0.18	10	638	151.83	495.831
0.83	0.17	10	638	151.83	503.729
0.84	0.16	10	638	151.83	511.627
0.85	0.15	10	640	162.69	519.597
⋮	Same Solution Values				⋮
0.99	0.01	10	640	162.69	631.973
0	1	12.5	571	51.92	-51.920
⋮	Same Solution Values				⋮
0.24	0.76	12.5	571	51.92	97.581
0.25	0.75	12.5	593	59	104.000
⋮	Same Solution Values				⋮
0.32	0.68	12.5	593	59	149.640
0.33	0.67	12.5	611	67.54	156.378
⋮	Same Solution Values				⋮
0.45	0.55	12.5	611	67.54	237.803
0.46	0.54	12.5	625	79.22	244.721
⋮	Same Solution Values				⋮
0.5	0.5	12.5	625	79.22	272.890
0.51	0.49	12.5	629	83.32	279.963
⋮	Same Solution Values				⋮
0.8	0.2	12.5	629	83.32	486.536
0.81	0.19	12.5	634	104.1	493.761
0.82	0.18	12.5	638	122.24	501.157
⋮	Same Solution Values				⋮

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
0.88	0.12	12.5	638	122.24	546.771
0.89	0.11	12.5	640	137.09	554.520
⋮	Same Solution Values				⋮
0.99	0.01	12.5	640	137.09	632.229
0	1	15	584	51.92	-51.920
⋮	Same Solution Values				⋮
0.26	0.74	15	584	51.92	113.419
0.27	0.73	15	605	59.5	119.915
⋮	Same Solution Values				⋮
0.38	0.62	15	605	59.5	193.010
0.39	0.61	15	627	72.22	200.476
0.4	0.6	15	610	62.67	206.398
⋮	Same Solution Values				⋮
0.43	0.57	15	610	62.67	226.578
0.44	0.56	15	618	68.92	233.325
⋮	Same Solution Values				⋮
0.51	0.49	15	618	68.92	281.409
0.52	0.48	15	629	80.63	288.378
⋮	Same Solution Values				⋮
0.62	0.38	15	629	80.63	359.341
0.63	0.37	15	633	87.24	366.511
⋮	Same Solution Values				⋮
0.79	0.21	15	633	87.24	481.750
0.8	0.2	15	640	114.9	489.020
⋮	Same Solution Values				⋮
0.99	0.01	15	640	114.9	632.451
0	1	17.5	590	51.92	-51.920

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
⋮	Same Solution Values				⋮
0.07	0.93	17.5	590	51.92	-6.986
0.08	0.92	17.5	606	53.2	-0.464
⋮	Same Solution Values				⋮
0.22	0.78	17.5	606	53.2	91.824
0.23	0.77	17.5	619	56.92	98.542
⋮	Same Solution Values				⋮
0.48	0.52	17.5	619	56.92	267.522
0.49	0.51	17.5	625	62.56	274.344
⋮	Same Solution Values				⋮
0.6	0.4	17.5	625	62.56	349.976
0.61	0.39	17.5	629	68.74	356.881
⋮	Same Solution Values				⋮
0.64	0.36	17.5	629	68.74	377.814
0.65	0.35	17.5	631	72.97	384.611
0.66	0.34	17.5	631	72.97	391.650
0.67	0.33	17.5	631	72.97	398.690
0.68	0.32	17.5	640	91.31	405.981
⋮	Same Solution Values				⋮
0.99	0.01	17.5	640	91.31	632.687
0	1	20	616	51.92	-51.920
⋮	Same Solution Values				⋮
0.24	0.76	20	616	51.92	108.381
0.25	0.75	20	631	56.92	115.060
⋮	Same Solution Values				⋮
0.29	0.71	20	631	56.92	142.577
0.3	0.7	20	635	58.61	149.473

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
⋮	Same Solution Values				⋮
0.72	0.28	20	635	58.61	440.789
0.73	0.27	20	638	66.62	447.753
⋮	Same Solution Values				⋮
0.85	0.15	20	638	66.62	532.307
0.86	0.14	20	640	78.2	539.452
0.99	0.01	20	640	78.2	632.818
0	1	22.5	638	51.92	-51.920
⋮	Same Solution Values				⋮
0.82	0.18	22.5	638	51.92	513.814
0.83	0.17	22.5	640	61.32	520.776
⋮	Same Solution Values				⋮
0.99	0.01	22.5	640	61.32	632.987
0	1	25	638	51.92	-51.920
⋮	Same Solution Values				⋮
0.8	0.2	25	638	51.92	500.016
0.81	0.19	25	640	59.65	507.067
⋮	Same Solution Values				⋮
0.99	0.01	25	640	59.65	633.004
0	1	27.5	640	51.92	-51.920
⋮	Same Solution Values				⋮
0.99	0.01	27.5	640	51.92	633.081
0	1	30	640	51.92	-51.920
⋮	Same Solution Values				⋮
0.99	0.01	30	640	51.92	633.081
0	1	32.5	640	51.92	-51.920
⋮	Same Solution Values				⋮

Coverage Weight	Distance Weight	Service Distance	Total Covered	Path Length	Objective
0.99	0.01	32.5	640	51.92	633.081
0	1	35	640	51.92	-51.920
⋮	Same Solution Values				⋮
0.99	0.01	35	640	51.92	633.081
0	1	50	640	51.92	-51.920
⋮	Same Solution Values				⋮
0.99	0.01	50	640	51.92	633.081