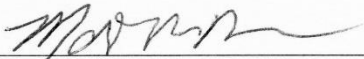
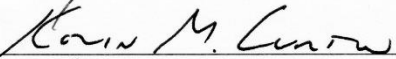
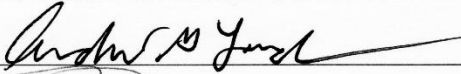

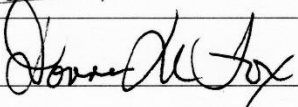
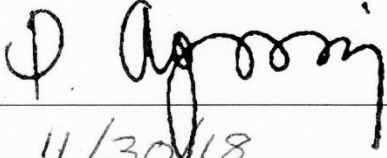


HEURISTICS TO APPROACH THE ORIENTEERING PROBLEM THROUGH NETWORK  
ANALYSIS IN GIS SOFTWARE

by

John Kip Robinson, Jr.  
A Thesis  
Submitted to the  
Graduate Faculty  
of  
George Mason University  
in Partial Fulfillment of  
The Requirements for the Degree  
of  
Master of Science  
Geographic and Cartographic Science

Committee:

	Dr. Matthew Rice, Thesis Director
	Dr. Kevin Curtin, Committee Member
	Dr. Andrew Loerch, Committee Member
	Dr. Dieter Pfoser, Department Chairperson
	Dr. Donna M. Fox, Associate Dean, Office of Student Affairs & Special Programs, College of Science
	Dr. Peggy Agouris, Dean, College of Science
Date: <u>11/30/18</u>	Fall 2018 George Mason University Fairfax, VA

Heuristics to Approach the Orienteering Problem through Network Analysis in GIS  
Software

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of  
Science at George Mason University

by

John K Robinson  
Master of Arts  
The George Washington University, 2010  
Bachelor of Arts  
University of Virginia, 2001

Director: Matthew Rice, Associate Professor  
Department of Geography and Geoinformation Science

Fall Semester 2018  
George Mason University  
Fairfax, VA

Copyright 2018 John K Robinson  
All Rights Reserved

## **DEDICATION**

This is dedicated to my family and friends who have supported me continuously as I plugged away at this project.

## **ACKNOWLEDGEMENTS**

I would like to thank the many friends, relatives, and supporters who have helped to make this happen. I would like to thank in particular Dr. Kevin Curtin, Professor in the Department of Geography at the University of Alabama and formerly of the Department of Geography and Geoinformation Science at George Mason University, for his tutelage, guidance, and inspiration over the last several years. His help made it possible to take a real-world problem and design a research study out of that experience. In addition, I would like to thank two faculty members at George Mason University: Dr. Matt Rice, Associate Professor in the Department of Geography and Geoinformation Science, for his guidance in completing the work; and Dr. Andrew Loerch, Associate Chair of the Department of Systems Engineering and Operations Research, for his support with Operations Research concepts and other assistance throughout. Finally, I would like to thank all of my professors from my time at George Mason for inspiring me and helping me to learn so much about geography and its myriad practices.

## TABLE OF CONTENTS

	Page
List of Tables .....	vii
List of Figures .....	viii
List of Equations .....	ix
List of Abbreviations and Symbols.....	x
Abstract .....	xi
1. Introduction and Research Question.....	1
2. Literature Review .....	5
2.1. Overview of Orienteering Problem Solution Approaches in the Literature .....	5
2.2. Problem Definition and Mathematical Formulation .....	9
2.2.1. Ensuring an Appropriate Problem Formulation.....	9
2.2.2. Unrooted Orienteering Problem.....	12
2.3. Developing GIS Software Heuristics for the TSP Family of Problems.....	14
3. Methods .....	17
3.1 Initial Heuristic Development.....	17
3.1.1. Nearest Neighbor Heuristic.....	19
3.1.2. Highest Available Heuristic .....	20
3.1.3. Heuristic Initialization .....	22
3.2. Finding Optimal Results .....	24
3.2.1. Microsoft Excel 2010 Solver Add-In.....	26
3.2.2. Gurobi Optimizer .....	30
4. Data.....	34
5. Results .....	38
5.1. Initial Rooted OP Heuristics .....	38
5.1.1. Initial Heuristic Results.....	38
5.1.2. Initial Heuristics with Initialization Results .....	41
5.1.3. Initial Heuristic Results Summarized .....	42
5.2. Unrooted OP Optimal Solutions .....	43
5.3. Example Implementation of Unrooted OP Heuristics .....	45
5.4. Unrooted OP Results: Heuristics Versus Optimal .....	48

6. Conclusions .....	53
7. Future Research Opportunities .....	58
Appendix A – Rooted OP Heuristics Definitions .....	63
Nearest Neighbor Heuristic .....	63
Highest Available Heuristic .....	64
Appendix B – Rooted OP Heuristics Implementation Output.....	67
Appendix C – Sample Optimal Model Gurobi Code.....	69
Appendix D – Iterative Process Identifying Total Cost Constraints.....	72
References .....	82

## LIST OF TABLES

Table	Page
Table 1: Heuristic Results on the Rooted OP Using Sets of All 66 Vote Locations and $C_{max}$ of 15,000 m .....	43
Table 2: Optimal Results for Dataset 10n_set2 with Progressively Smaller $C_{max}$ Values	44
Table 3: Results of Implementation of NNH on Dataset 10n_set2 with $C_{max}$ of 4,696 m.	45
Table 4: Results of Implementation of HAH on Dataset 10n_set2 with $C_{max}$ of 4,696 m.	47
Table 5: Heuristic Performance Relative to Optimal Results for the Unrooted OP .....	49
Table 6: Means, Standard Deviations, and 95% Confidence Intervals for Percentage of Optimal Measurements for Each Heuristic .....	50
Table 7: Means, Standard Deviations, and 95% Confidence Intervals for Percentage of Optimal Measurements for Each Heuristic, by Dataset Size .....	51
Table 8: Statistical Summary of Difference in Percentage Optimal Measurements between Heuristics .....	52
Table 9: Statistical Summary of Difference in Percentage Optimal Measurements for Heuristics Found Significantly Different .....	52



## LIST OF FIGURES

Figure	Page
Figure 1: Screenshot of the Excel model for a 5-vertex problem—constraint 5 is built into the sheet in cells below this screenshot.....	27
Figure 2: The Excel Solver Add-In pop-up window seen on right showing the objective cell, decision variable cells, and most of the constraints of the model. ....	29
Figure 3: First part of a 10-vertex LP-format model run by the Gurobi Optimizer at the command line. ....	32
Figure 4: The complete city of Fairfax, Virginia road network and 66 vote location sites. ....	35
Figure 5: Solution path found by the Nearest Neighbor Heuristic for rooted OP set 1....	39
Figure 6: Solution path found by the Highest Available Heuristic for rooted OP set 1. ..	40
Figure 7: Solution path found by the Nearest Neighbor Heuristic for unrooted OP dataset 10n_set2 with $C_{max}$ of 4,696m. ....	46
Figure 8: Solution path found by the Highest Available Heuristic for unrooted OP dataset 10n_set2 with $C_{max}$ of 4,696m. ....	47

## LIST OF EQUATIONS

Equation	Page
Equation 1: Unrooted Orienteering Problem Formulation .....	13

## LIST OF ABBREVIATIONS AND SYMBOLS

$\Delta$ .....	Change in
Ant Colony System .....	ACS
Attractive Traveling Salesman Problem .....	AtTSP
Biobjective Shortest Path Problem .....	BSP
Geographic Information System .....	GIS
Highest Available Heuristic .....	HAH
Highest Available Heuristic with Initialization .....	HAHwI
Integer Linear Programming .....	ILP
Linear Programming .....	LP
Mixed Integer Programming .....	MIP
Nearest Neighbor Heuristic .....	NNH
Nearest Neighbor Heuristic with Initialization .....	NNHwI
Orienteering Problem .....	OP
Orienteering Problem with Mandatory Visits and Conflicts .....	OPMVC
Orienteering Problem with Time Windows .....	OPTW
Operations Research .....	OR
Orienteering Tour Problem .....	OTP
Selective Traveling Salesman Problem .....	STSP
Time-Dependent Orienteering Problem .....	TD-OP
Team Orienteering Problem .....	TOP
Traveling Salesman Problem .....	TSP

## **ABSTRACT**

### **HEURISTICS TO APPROACH THE ORIENTEERING PROBLEM THROUGH NETWORK ANALYSIS IN GIS SOFTWARE**

John K Robinson

George Mason University, 2018

Thesis Director: Dr. Matthew Rice

The orienteering problem (OP) is a subclass of the traveling salesman problem (TSP) that includes a constraining function on the solution path that may restrict the number of vertices visited and a requirement to maximize the value of the vertices visited. Solving the OP is often approached by using linear programming (LP) and branch-and-bound algorithms. Existing research on the OP does not provide algorithms or heuristic approaches for solving this class of problems using only geographic information system (GIS) software and a spreadsheet. Several new heuristic approaches are presented and tested on small and medium real-world road network datasets. The heuristics are evaluated relative to the optimal solution found through an integer linear programming (ILP) approach. Heuristics evaluated include the Nearest Neighbor Heuristic (NNH) and Highest Available Heuristic (HAH). A centralizing initialization step is also introduced and evaluated. The new heuristic approaches make it possible to find a good solution for

a small OP in network space without requiring knowledge of or access to LP software.

Future research is called for to develop more effective heuristics, further validate existing heuristics, and improve the implementation times of existing heuristics.

## **1. INTRODUCTION AND RESEARCH QUESTION**

The traveling salesman problem (TSP) is one of the most studied problems in combinatorial optimization. Within this broad category of problem type exist multiple subclasses of problems, one of which is known by several names in the literature—most prominently as the orienteering problem (OP). In the OP, each vertex has a score, and the goal is to maximize the value of the vertices visited while staying within a defined cost constraint. Similar to the TSP, each vertex has a set distance between it and all other vertices, and so the vertices are typically treated as sitting on a defined network. The cost constraint of the OP can be operationalized in many ways such as travel cost or time, but is often defined as travel distance.

The name for the orienteering problem is derived from the sport of orienteering, wherein competitors using a map and compass work to navigate a set of points. The sport of orienteering itself arose from military training exercises of land navigation. According to Orienteering USA, the national governing body for the sport in America, there are four internationally sanctioned formats of orienteering competition: foot orienteering, trail orienteering, ski orienteering, and mountain bike orienteering (Orienteering USA, 2018). Other styles include adventure racing, canoe orienteering, radio orienteering, and rogaining.

Foot orienteering is the classic format of the sport and somewhat resembles the TSP: competitors seek to visit all control points as quickly as possible, usually with a set start and end point. While the TSP is frequently placed in network space, foot orienteering is different in that it is practiced in planar space and in some cases the order of points to visit is defined (Orienteering USA, 2018). Ski and mountain bike orienteering also involve visiting an ordered set of points as quickly as possible, but these disciplines usually require staying on a defined network of paths. Rogaining is a long-distance and cross-country form of competitive orienteering practiced in small teams that introduces scored checkpoints. (The name for the sport was coined from blending parts of the names of the three inventors: Rod, Gail, and Neil (Victorian Rogaining Association, 2016).) A variation of foot orienteering that incorporates scored control points is known as Score-O. Rogaining courses are typically designed so that teams will not be able to visit all checkpoints within the time limit (anywhere from 2 to 24 hours) (International Rogaining Federation, 2017). Thus rogaining teams must strategically plan their route to accumulate as many points as possible from the checkpoints they do visit. Rogaining, like foot orienteering, is also conducted in planar space rather than network space. This aspect is important because the landscape can vary dramatically with hills, rivers, or canyons, and so the shortest “distance” between two points may be based on time to traverse instead of straight-line distance. This also means that the shortest “distance” between two points in rogaining may vary among competitors. With respect to specific orienteering competition types, the OP can best be thought of as rogaining but conducted in more restricted network space like ski and mountain bike orienteering.

The OP can have numerous potential applications beyond an orienteering competition, such as a traveling politician attending a number of rallies where he or she will attempt to secure the votes of the attendees at each differently-sized event, or similarly for a college recruiter attending high schools seeking to accumulate applications at each stop. Another variation of the problem could be as an inventory and delivery routing problem for a single vehicle, such as the delivery of home heating fuel. The OP has also been formulated as a prize-collecting TSP scenario. Extensions of the OP have been studied that may apply to school bus routing, industrial refuse collection, and dial-a-ride services. In addition, as the orienteering problem is a subclass of the traveling salesman problem, the team orienteering problem (TOP) is a subclass of the vehicle routing problem, with its own body of research.

A number of solution techniques have been proposed for various formulations of the OP, as will be outlined in chapter 2. However, there appears to be a paucity of research into heuristics that can be applied using only geographic information system (GIS) software. Given the large array of real world applications of variations of the OP and the complexity typically involved in the implementation of a linear programming approach (access to specialized software and the programming ability required to use such software), there is a need for more accessible solution approaches to the OP for GIS users of varying expertise. While heuristic techniques using a GIS software package may not match the optimality achieved by linear programming (LP) or other complex programming approaches in every case, such techniques may be more accessible for many users with access only to GIS software and a spreadsheet or limited training on



complex programming approaches. Solution times with heuristics in the GIS may be significantly shorter than those experienced with optimal solution procedures.

The goal of this study will be to identify the best heuristic possible, or at least to suggest one or more good heuristics, for application to the OP with GIS software and a spreadsheet. The rest of this thesis is organized as follows: in chapter 2 the literature of the orienteering problem is reviewed and the mathematical formulation for the OP is specifically defined. In chapter 3 the methods employed in the study are outlined, including heuristic formation and finding optimal solutions for problem sets. Chapter 4 describes the data to be used in the study and chapter 5 describes the results found, including examples of successful implementation of the heuristics and that the average percent optimal achieved by the Highest Available Heuristic (HAH) and HAH with Initialization (HAHwI) were higher than that of the Nearest Neighbor Heuristic with Initialization (NNHwI) to a statistically significant degree. Chapter 6 outlines the conclusions derived from the work and chapter 7 adds future areas of research that may be suggested by the results and conclusions of the study.

## **2. LITERATURE REVIEW**

What follows is an overview of the academic research on the Orienteering Problem. Additionally, a presentation of the mathematical formulation for the OP is provided along with an explanation for why the unrooted variant of the OP was selected for this research over the rooted variant of the problem. Finally, research related to the utilization of heuristics in a GIS environment is briefly discussed.

### **2.1. Overview of Orienteering Problem Solution Approaches in the Literature**

The orienteering problem has been approached in a number of ways that utilized some type of computer programming approach. One of the early research efforts was from Tsiligirides (1984) who developed two heuristics to solve the OP. The first was called the stochastic algorithm (S-algorithm), which used a Monte Carlo method to create many possible routes and a desirability factor and then pick the best route. The second heuristic was called the deterministic algorithm (D-algorithm), which built on a vehicle routing concept that utilized two concentric circles to identify the next possible point to add to the route. A vertex switching post-processor module was also added to improve the results of both heuristics (Tsiligirides, 1984). Kataoka and Morito (1988) employed a branch-and-bound algorithm to solve a relaxed variant of the OP (deemed the single constraint maximum collection problem there) that may include subtours sometimes precluded in some definitions of the parent problem. Laporte and Martello (1990) called this problem the Selective TSP (STSP), and employed a branch-and-bound algorithm to gradually extend the path through a breath-first branch and bound process. They also

used two heuristics—a nearest neighbor greedy heuristic and a cheapest insertion heuristic to provide initial feasible solutions for the branch-and-bound algorithm (Laporte & Martello, 1990).

Another approach for the OP was provided by Ramesh and Brown (1991), who used a four-phase process. The first phase used a single-point and then double-point insertion procedure to build an initial path. The second phase used a 2-opt and then 3-opt procedure for each edge along the path. Phase three deleted various vertices to allow more room under the budget constraint for the future addition of more points. After phases 1–3 were performed iteratively, in phase four relatively lower value points that might have been missed thus far were checked for addition to the solution path (Ramesh & Brown, 1991). Chao, Golden, and Wasil (1996) presented a heuristic that was initialized by installing an ellipse over the vertices with the start and end points as foci and the cost constraint set as the long axis. A path was then constructed using a greedy cheapest-insertion concept until the cost constraint was hit. Additional paths were constructed in this way from the remaining available points in the ellipse. The highest scoring path was the initial path for the heuristic. The improvement step of the heuristic utilized a two-point switch mechanism using a greedy cheapest-switch concept. The third step was a single-point insertion step to add any additional available points not on the currently best path, again using a greedy concept. The authors found their heuristic to perform as well or better than seven other heuristics and algorithms in the literature (Chao et al., 1996). Gendreau, Laporte, and Semet (1998) developed a tabu search heuristic process to solve the OP, or STSP as it is called there. This tabu search initialized

with their *Insert and Shake* heuristic and then used their GENIUS program tour construction and post-optimization phase to begin building improved solution paths. Additional steps in the solution process included identifying potential clusters of points to add to the tour, iterating through potential moves while periodically increasing the importance of the length of the tour and profits collected, and again applying the GENIUS process at regular intervals. This tabu heuristic was found better and more stable than other contemporary heuristics (Gendreau et al., 1998).

Several solution approaches to variations of the OP also appear in the literature. Righini and Salani (2009) presented a dynamic programming algorithm to solve the OP with Time Windows (OPTW). This variation of the OP stipulates that some vertices can only be visited to collect a score during certain time windows; this formulation can apply to bus route planning and repair technician scheduling. The algorithm used a technique called decremental state space relaxation to develop a solution. The problem was relaxed to initially allow multiple location visits while defining a set of critical vertices that can be visited only once. All vertices were labeled, solution paths were tested bidirectionally, and several existing point insertion techniques were applied (Righini & Salani, 2009). Verbeeck, Sörensen, Aghezzaf, & Vansteenwegen (2014) provided a solution method for the time-dependent OP (TD-OP), in which the travel time (edge cost) between two vertices varied depending on the time of day. This problem modeled congestion related factors found in rush hour and off-peak use on a road or other network types, and was presented in a mixed integer programming (MIP) formulation. The solution implemented a metaheuristic based on an ant colony system (ACS), which used an insert local search

procedure, a modified 2-opt procedure on edges, and the ACS framework to incrementally construct multiple solution paths that were increasingly greedy. This method performed very successfully with a low average result gap relative to the optimal result (Verbeeck et al., 2014). Other variations on the OP found in the literature included the OP with compulsory vertices; and the capacitated OP, which allowed for additional constraints to the OP such as vehicle capacity.

More complex formulations of the OP have been addressed as well. Duque, Lozano, and Medaglia (2015) presented a solution algorithm for the biobjective shortest path problem (BSP), which is similar to the OP except that the primary cost/distance constraint is separated into two variables, typically noted as time and cost. The pulse algorithm, which was programmed to search forwards and backwards through possible solution paths while incrementally pruning inefficient paths, was found to be effective and more efficient for the BSP than existing label-setting techniques. In addition, the pulse algorithm was also able to find a good approximation of the most optimal path very quickly and was cited for possible use as a heuristic in time-restricted settings (Duque et al., 2015). Erdoğan, Cordeau, and Laporte (2010) addressed a variation of the TSP called the Attractive TSP (AtTSP), where each vertex on the graph was assigned a fractional attractiveness value representing the ratio of potential customers drawn to each facility. In this research, after the attractiveness probability was described as a variation on an existing gravity function in the literature, a non-linear integer programming formulation was provided to describe the AtTSP. For solution approaches, a branch-and-cut algorithm and tabu search heuristic were proposed and implemented (Erdoğan et al., 2010).

In addition to this review, three surveys of the OP and related problems have been published in recent years. The paper by Vansteenwegen, Souffriau, and Oudheusden (2011) covered research on various OP and TOP problems and serves as a useful introduction to the problem. Feillet, Dejax, and Gendreau (2005) called this class of problems TSP with profits, and sorted much of the research up to that time into several classifications by specific problems and by solution approaches. Gunawan, Lau, & Vansteenwegen (2016) provided a survey of more than 80 recent research papers on OP variants, solution approaches, and modern practical applications.

## **2.2. Problem Definition and Mathematical Formulation**

With numerous variants to the OP as well as slightly different formulations for the general OP found in the literature, the exact formulation used in this research must be specified before any heuristics can be evaluated. The intention for this research was always to work with the general OP formulation, with OP variants reserved for possible future research. However, one factor that needed to be determined was whether to use a rooted or unrooted variant of the general OP—that is whether the problem would require finishing at the same start vertex (rooted) or would permit finishing at a separate end vertex (unrooted).

### **2.2.1. Ensuring an Appropriate Problem Formulation**

Research and experimentation was undertaken to select for the study an OP formulation that would be appropriate and correct for the problem type used. The OP type envisioned originally for this research was the rooted—or circuit—variant, wherein the solution path is required to return to the start vertex. The first heuristics created and

tested were implemented using the rooted OP concept. However, during optimal solution procedure development, problems and errors arose in testing the model and the specific formulation used could not be eliminated as a source of problems. As a result, a further review of OP literature was conducted specifically focusing on research that included a written formulation of the general OP. This was done with the goal of identifying a formulation that would best facilitate a successful optimal solution procedure development process. Both rooted and unrooted formulations for the OP were considered during this review, though the rooted OP was preferred since it was used in the initial heuristic development.

The first paper identified with a math formulation for the OP was Kataoka and Morito's paper in 1988. This work provided two formulations for what can be considered the rooted OP. The first was a formulation that included edges and vertices, and the second a Lagrange-relaxed variant that included self-loops at each vertex (Kataoka & Morito, 1988). The Laporte and Martello research of 1990 used an integer decision variable (rather than a binary variable) for its bidirectional graph variant of the problem. Their subtour elimination/connectivity constraints used set notation not typically found in other OP research. A paper by Ramesh, Yoon, and Karwan (1992) also included a formulation for the rooted OP, which was called the orienteering tour problem (OTP) in that case. This solution approach assigned all vertices in the complete graph to either the solution subgraph or a subgraph of dummy edges for unvisited vertices, which was deemed likely to be quite labor intensive to model for application to this research.

In addition to early OP literature, several recent papers that explored extensions to and variants of the OP also provided written formulations of the problem. A paper by Palomo-Martínez, Salazar-Aguilar, and Albornoz (2017) that explored an OP variant known as the OP with Mandatory Visits and Conflicts (OPMVC) also provided analysis of various subtour elimination approaches to OP formulations in the literature. The approach that used a single-commodity flow formulation to avoid subtours was found to solve to optimality within one hour more often than the other formulations. While this finding is of note and should be considered for future work, the single-commodity flow subtour elimination constraint formulation was not chosen for implementation in this research.

Another option was provided in the survey of the OP literature by Vansteenwegen et al. (2011). A survey was identified as a reliable source because the authors reviewed multiple formulations in the literature and published a version of the OP formulation that used relatively modern and standardized notation. The formulation published in this survey used the unrooted variation of the OP.

While this review of the OP literature and formulations was pursued, development of an optimal solution model was ongoing at the same time. In addition to experimenting with approaches to model the rooted OP, development of an optimal solution model for the unrooted OP formulation laid out in Vansteenwegen et al. (2011) was also attempted. Ultimately, through trial and error, the first model built successfully to find an optimal solution for the OP used the unrooted OP variant, and thus the research shifted to applying and evaluating the developed heuristics to this particular formulation of the OP.



### 2.2.2. Unrooted Orienteering Problem

The unrooted formulation for the OP can be understood as follows: Let  $V$  be the set of vertices (or locations) such that  $V = \{v_1, \dots, v_n\}$  and  $E$  be the set of bidirectional edges in between the points in  $V$ . The complete graph of the vertices and edges is  $G = \{V, E\}$ . Each edge ( $e_i$ ) is assigned a cost ( $c_{ij}$ ), which is typically a symmetric nonnegative value associated with the distance, travel time, cost, or some other function of traveling from  $v_i$  to  $v_j$ . The start point is vertex 1 ( $v_1$ ) and the end point is vertex  $n$  ( $v_n$ ). (To make the problem into a rooted variation of the OP, the end point is also set to vertex 1.) Each vertex  $v_j$  has a score ( $s_i$ ) that is nonnegative (some variations of the OP assign a negative penalty score to a vertex under certain conditions). The start vertex  $v_1$  and end vertex  $v_n$  are given a score of zero. The scores  $s_i$  are completely additive and each vertex  $v_j$  can only add to the total score  $S$  once in the solution path. The goal is to maximize the total score  $S$  while keeping the total travel cost function under the preset value  $C_{max}$  by finding the Hamiltonian path  $G'$  over some subset of  $V'$  of the total set of vertices  $V$ . The two primary tasks required in the OP have been called vertex selection and finding the shortest Hamiltonian path between the selected vertices (Vansteenwegen et al., 2011).

The unrooted OP can be illustrated using the following mathematical formulation (Equation 1). Two decision variables are used in the formulation below: the binary variable  $x_{ij} = 1$  if a stop at vertex  $i$  is followed by a stop at vertex  $j$ , and  $x_{ij} = 0$  if not; and the integer variable  $u_i$ , which assigns the relative position of the vertex in the path.

**Equation 1: Unrooted Orienteering Problem Formulation**

$$\text{Maximize } \sum_{i=2}^{n-1} \sum_{j=2}^n S_i x_{ij}, \quad (0)$$

*subject to*

$$\sum_{j=2}^n x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1, \quad (1)$$

$$\sum_{i=1}^{n-1} x_{ik} = \sum_{j=2}^n x_{kj} \leq 1; \forall k = 2, \dots, n-1, \quad (2)$$

$$\sum_{i=1}^{n-1} \sum_{j=2}^n c_{ij} x_{ij} \leq C_{max}, \quad (3)$$

$$2 \leq u_i \leq n; \forall i = 2, \dots, n, \quad (4)$$

$$u_i - u_j + 1 \leq (n-1)(1 - x_{ij}); \forall i, j = 2, \dots, n, \quad (5)$$

$$x_{ij} \in \{0,1\}; \forall i, j = 1, \dots, n; i \neq j. \quad (6)$$

The objective function (0) seeks to maximize the total collected score  $S$ . The remaining equations represent the constraints that define the OP. Constraint (1) initializes the path by ensuring vertex  $v_1$  is part of the solution path  $G'$  and that the solution path finishes at vertex  $v_n$ . The constraints in (2) are the flow conservation constraints that ensure the path is fully connected and that the vertices are visited no more than once on the solution path. Constraint (3) is the total travel cost function constraint. Constraints (4) and (5) ensure that no subtours are included in the solution path, and are formulated per the Miller-Tucker-Zemlin TSP formulation (Miller, Tucker, & Zemlin, 1960). Constraint

(6) defines the decision variable  $x_{ij}$  as binary and that the path cannot exit a vertex and return to that same vertex in a loop.

### **2.3. Developing GIS Software Heuristics for the TSP Family of Problems**

Limited research has been conducted relating to the development of heuristic solution approaches for the TSP family of problems that utilize just GIS software; most approaches utilize some form of linear programming. The primary factor driving the use of such approaches is that the OP has been shown to be NP-hard (Gendreau et al., 1998; Golden, Levy, & Vohra, 1987; Laporte & Martello, 1990). Since the goal of operations research (OR) frequently is to find optimal solutions to combinatorially complex problems, researchers naturally seek solution approaches that offer the most optimal results in the most efficient manner. However, since there are many real world applications of the OP but also many challenges involved in the implementation of a linear programming approach (such as specialized software and advanced programming capabilities), there is a need for more approachable solution techniques to the OP for GIS users of varying expertise. Church (2002) described the prevalence of GIS software packages in public and private settings and several ways that GIS can be used to find solution approaches to some GIS-Location problems.

With respect to research related to GIS-based solutions for the TSP family of problems, Curtin, Voicu, Rice, & Stefanidis (2014) showed that four undisclosed solution approaches for the general TSP programmed into GIS software packages by Esri and Intergraph must be heuristics because each approach failed to find known optimal solutions for several cases of between 12 and 20 points. This finding was consistent with

the work of Church and Sorenson (1994), which made it clear through the example of the p-median problem that tackling complex location science problems is challenging for GIS software, especially because of the issue of solution approaches becoming stuck in local optima. In Curtin et al. (2014), the heuristics by Esri and Intergraph provided solutions that were found to be up to 14% suboptimal, and even the optimal solutions for smaller datasets could not be guaranteed to be optimal by the software packages.

Concepts employed from other areas of research may be appropriated and used in the development of heuristics to approach the OP. One study of participants from orienteering competitions with a defined end point found that more experienced participants built solution paths in a backwards-looking manner, whereas inexperienced participants built forward-looking paths (Eccles, Walsh, & Ingledew, 2002). Developing heuristic concepts that look throughout the dataset before adding vertices to the solution path may prove fruitful and improve the efficiency of the heuristic. In addition, Cao, Sun, & Macleod (1999) implemented a GIS platform serving as a user interface in conjunction with a computer program operating in the background to solve TSP-type problems after user input was provided to define the problem parameters. The GIS interface did not conduct any of the problem-solving, but significantly eased the user's interaction with the problem-solving structure. This concept may suggest an approach to solve problems such as the OP with sufficient power while maintaining a relatively lower bar of expert knowledge required to access the problem-solving technique.

While the research literature on the OP has grown and developed dramatically since 1984, no research was found specific to development of solution approaches for the

OP that has been implemented entirely in GIS software. In addition, there is limited research into practical solution approaches for the broader category of the TSP family of problems that can be implemented using GIS software and a spreadsheet and without complex LP. This research is an effort to begin to address this gap in the current literature.

### **3. METHODS**

The methods employed in this study to develop and evaluate the heuristics utilized were constructed and refined through iterative processes. Early work included developing the main heuristics with inspiration from concepts in geospatial analysis. Later work focused on developing a process to calculate the optimal solution for each problem set in order to be able to evaluate the relative merits of each heuristic in comparison with the optimal solution. These methods are enumerated in the subsections below.

#### **3.1 Initial Heuristic Development**

During the initial period of heuristic development, ArcMap 10.2.2 for Desktop with Network Analyst extension was employed on a Dell OptiPlex 9010 Intel® Core™ i7-3770 CPU running Windows 7 Enterprise. Sixty-six address locations in the city of Fairfax, Virginia were geocoded to the city of Fairfax, Virginia road layer, and a network dataset was constructed using this address and road layer information. In order to build a matrix of all possible trip lengths from one location to each of the others, the OD Cost Matrix tool through the Network Analyst toolbar was run for all 66 locations.

During initial heuristic development the rooted OP concept was used. It was a simple task later to modify the initial heuristics to fit the unrooted OP variant. (The initial rooted OP heuristics can be found in Appendix A.) For initial development and heuristic testing, the complete set of 66 points was used. The total cost constraint of the OP (distance is the cost function in this study) was set at 15,000 meters (m), which was just

over half the mean minimum distance needed to travel a complete circuit of all 66 locations to include starting from and returning to the root location; this mean minimum circuit distance was 29,158 m. The mean minimum distance needed to travel a complete circuit of all locations including returning to the root location was computed by using the New Route tool available through the Network Analyst toolbar to find the minimum distance to travel a complete circuit with each of the 66 locations set as the root location. The total cost constraint can be adjusted to create further variations of problem instances on which to test the heuristics without requiring additional new datasets. We know from Curtin et al. (2014) that this function in ArcMap 9.1 returned results up to 14% lower than the optimal path found through an LP process; however, this function was the best known method available at the time of initial heuristic development to find this distance.

The heuristics presented in the following sections are framed in terms of a traveling politician attending rallies and gathering all of the votes available at each location at which she stops. These heuristics use standard built-in ArcGIS tools that should be straightforward to use for any intermediate user. They are presented below in a format conducive to manual implementation by any user, but different options exist for automating some or all parts of their implementation, including the ArcGIS ModelBuilder environment. The heuristic procedures were tested on two sets of the 66 address locations with different root locations and randomized vote values at all locations. After a procedure to find optimal solutions for problem instances was developed, the initial heuristics were converted to address the unrooted variant of the OP.

### 3.1.1. Nearest Neighbor Heuristic

A relatively simple heuristic to implement is the Nearest Neighbor Heuristic (NNH). This path construction heuristic starts at the start vertex  $v_1$ , finds the nearest location on the network that is not already in the solution path, and continues to add locations while checking to make sure the path includes room under the total cost constraint for the trip to the end vertex  $v_n$  to complete the path.

Nearest Neighbor Heuristic:

Step 1: Start at the start vertex  $v_1$ .

Step 2: Find the nearest available vote location.

- a. Using the OD Cost Matrix, find the trip length to the nearest available undiscarded vote location  $v_j$  that is not already on the solution path and check to see if the point is within the total cost constraint  $C_{max}$ .
- b. If there are no unselected and undiscarded destinations  $v_j$  available within the total cost constraint  $C_{max}$ , proceed to Step 3.
- c. Ensure that there is sufficient distance remaining in the total cost constraint  $C_{max}$  to travel to the end vertex  $v_n$  from the proposed destination. If there is not sufficient distance remaining, the destination cannot be selected and is discarded for this iteration. Return to Step 2(a).
- d. Compute the distance traveled on the trip.
- e. Add the distance traveled on the trip to the total distance traveled, and the votes obtained on the trip to the total votes.



f. Proceed from the selected destination to Step 2(a).

Step 3: Once it is shown that no more vote locations  $v_j$  can be added to the trip without violating the total cost constraint  $C_{max}$ , including the distance needed to travel to the end vertex  $v_n$ , the final step is simply to proceed to the end vertex  $v_n$ . The distance of this final trip is added to the total distance traveled subtotal to confirm that the total trip has remained under the total cost constraint  $C_{max}$ . No additional votes are added at the end vertex  $v_n$ .

Each tested destination was catalogued in a spreadsheet to track all visited and rejected destinations, total votes accrued, and the total distance traveled. When testing the rooted OP variant of the NNH, additional vote destinations located on the final path back to the root were checked for in Step 3 using the Find Route function in ArcGIS to draw the shortest path between the last point on the path and the root location.

### **3.1.2. Highest Available Heuristic**

The Highest Available Heuristic (HAH) uses many of the basic mechanics of the Nearest Neighbor Heuristic, but is built as a greedy heuristic by searching iteratively for the highest scoring location available.

Highest Available Heuristic:

Step 1: Start at the start vertex  $v_1$ .

Step 2: Find the highest available vote location.

a. Select the highest available undiscarded vote location  $v_j$ .

- b. Using the OD Cost Matrix, find the trip length to the highest available undiscarded vote location  $v_j$  and check to see if the point is within the total cost constraint  $C_{max}$ .
- c. If there are no unselected and undiscarded destinations available within the total cost constraint  $C_{max}$ , proceed to Step 3.
- d. Ensure that there is sufficient distance remaining in the total cost constraint  $C_{max}$  to travel to the end vertex  $v_n$  from the proposed destination. If there is not sufficient distance remaining, the destination cannot be selected and is discarded for this iteration. Return to Step 2(a).
- e. Select and note all additional unvisited destinations that are located on the trip.
- f. Compute the distance traveled on the trip and total votes obtained at all destinations.
- g. Add the distance traveled on the trip to the total distance traveled, and the votes obtained on the trip to the total votes.
- h. Proceed from the selected destination to Step 2(a).

Step 3: Once it is shown that no more locations can be added to the trip without violating the total cost constraint  $C_{max}$ , including the distance needed to travel to the end vertex  $v_n$ , check the trip from the last destination on the path to the end vertex for any additional destinations that have not been added to the total trip thus far. Add the vote

totals from any locations on the final trip to the end vertex to the overall total votes accumulated.

Again, each tested destination was catalogued in a spreadsheet to track all visited and rejected destinations, total votes accrued, and the total distance traveled. To identify visually all additional destinations located on a tested trip, the Find Route function in ArcGIS was used to draw the shortest path between the tested origin and destination points.

### **3.1.3. Heuristic Initialization**

The Nearest Neighbor and Highest Available Heuristics described above initialize from the start vertex and proceed directly into path-building. The performance of these heuristics could be significantly affected by the geography of the underlying network and vertex locations, especially the start vertex of the dataset. An outlying start vertex is more likely to affect the NNH since the solution path is built by adding the next closest location to the path without consideration of the broader result.

One option to address possible reductions in performance by the heuristics due to the geography of the data is to introduce a centralizing initialization step to the heuristic process. To mitigate the potential effect of a remote start vertex or outlying points on the heuristics, a simple initialization step to find the 1-median of all locations in the dataset was implemented with both heuristics. The Location-Allocation tool available through the Network Analyst toolbar in ArcGIS was set to solve the problem type Minimize Impedance for 1 facility (the 1-median of the dataset on the network is the “facility” in this tool) in order to find the 1-median of the locations in the dataset. This initialization

step was first tested on the rooted OP variants of the heuristics, and has been modified below to apply to the unrooted OP heuristics.

The initialization step can be inserted as a new step 2 to both the NNH and HAH, moving their original step 2 to step 3:

Step 2: Find the 1-median of all vote locations.

- a. Identify the 1-median of vote locations through Location-Allocation analysis.
- b. If the 1-median vote location is also the start vertex, proceed to step 3.
- c. Select the 1-median vote location or other central location.
  - i. Using the OD Cost Matrix, ensure that the 1-median vote location is within the total cost constraint  $C_{max}$ .
  - ii. If the 1-median vote location is not within the total cost constraint  $C_{max}$  then the initialization step may not be efficient for the solution path.
  - iii. If a central location is still desired for initialization, use the OD Cost Matrix to find the location nearest to the 1-median and check to see if its distance to the start vertex is within the total cost constraint  $C_{max}$ . Proceed iteratively through the locations closest to the 1-median until one is identified that is within the total cost constraint  $C_{max}$  to the start vertex.

- iv. Ensure that there is sufficient distance remaining in the total cost constraint  $C_{max}$  to travel to the end vertex from the 1-median location (or other central location selected).
- d. Select and note all additional destinations that are located on the trip from the start vertex  $v_1$  to the 1-median or other central location selected for inclusion on the solution path.
- e. Compute the distance traveled on the trip and total votes obtained at all destinations.

Initializing the heuristics by first moving to the 1-median of the dataset is designed to quickly move the solution path closer to more candidate points. This initialization step will not always improve the performance of a heuristic, depending on the underlying network and location data. It may be beneficial to apply the heuristics with and without the centralizing initialization step to find the best possible result.

### **3.2. Finding Optimal Results**

After the development of two heuristics and one initialization step, it was necessary to develop a procedure that would find the optimal solution (or one of multiple optimal solutions in some cases) to the problem instances. In order to assess the performance of the heuristics presented in this study, the optimal solution path for each dataset would have to be determined in order to provide a point of comparison for the heuristics.

Optimal solution procedure development was conducted using ArcMap versions 10.3.1 and 10.6.1 on a Dell XPS L502X Intel® Core™ i7-2670QM CPU running

Windows 7 Home Premium. As has been consistently shown in the literature, in general terms the best approach for solving the OP and other problems from the TSP family is to employ an LP approach. Powerful software suites for application to optimization problems using LP include the Mathematical Programming Language (MPL) Modeling System from Maximal Software used in conjunction with IBM's CPLEX solver, and the Gurobi Optimization math programming solver. Before proceeding to develop a full LP model able to handle small to large problem sizes, an initial effort to build a working model for small problems was made using a Microsoft Excel 2010 spreadsheet and the Excel Solver Add-In built by Frontline Solvers, Inc. Since Excel and the Excel Solver Add-In were more familiar than any of the LP solver software packages, by first developing a model in a spreadsheet setting it would facilitate the process of ensuring that the model and the basic model parameters were built correctly.

After developing a model to solve a small theoretical example of the OP, the next goal was to scale that model up incrementally to optimally solve a representation of a small to moderate-size real world problem such as that presented by the Fairfax city data in this study. However, scaling up the model was not a linear process since the model was first built in an Excel spreadsheet for validation and then was expanded by coding the model into an LP-format file to be run at the command line by the Gurobi Optimizer. The next sections describe the processes of building the solver model in Excel and then in an LP-format file.

### 3.2.1. Microsoft Excel 2010 Solver Add-In

The first effort to build a working model in Excel was intended to be built with a rooted OP and ten random points chosen from the Fairfax City dataset of 66 points (this dataset is described in chapter 4). Each of the ten points randomly was assigned a score between 1 and 100. Before the ten-vertex model was completed, it became clear during early testing that the problem was too large for the native Excel Solver Add-In to handle. The native Excel Solver is limited to 100 constraints and 200 decision variables, and the number of constraints in the model needed to solve a ten-vertex problem was larger than that limit. The test problem was shrunk down by randomly selecting five of the original ten points in order to build a model that would fit under the constraints and decision variables limits. Basic aspects of the Excel model included: a binary decision variable cell matrix corresponding to every  $i, j$  pair and where the Solver would assign a value of 1 to all vertices included in the solution path; a cell matrix of the distances between all vertices; an objective function cell that added the score of each vertex included in the solution path; and cells to calculate or otherwise handle most of the constraints in the model. See Figure 1 for a basic layout of the model components in an Excel sheet.

Multiple approaches to solving the first rooted OP model formulation on the five-point dataset using Excel Solver were constructed. During this model development process, it was necessary to work through issues with handling correctly several of the model's constraints. Building into the model the constraints to handle path initializing and the total travel cost was straightforward. To initialize the solution path, it was necessary to ensure that the sum of all the binary decision variable cells in the row where

$i = 1$  (except for  $j = 1$ ) was set to a total of 1. For the total travel cost function constraint  $C_{max}$ , the binary decision variable cells were multiplied by the corresponding distance cells and added together using Excel's SUMPRODUCT function. The first constraint to cause issues was the flow conservation constraint, which ensured that the solution path stopped at each vertex no more than one time per vertex. This constraint was handled by ensuring that each row (not including the initial vertex) and column (except for  $j = 1$  and not including the last vertex of the path) of binary decision variable cells summed to 0 or 1, and that the sum for matching  $i$  and  $j$  values was equal for all values from 2 to  $n - 1$ .

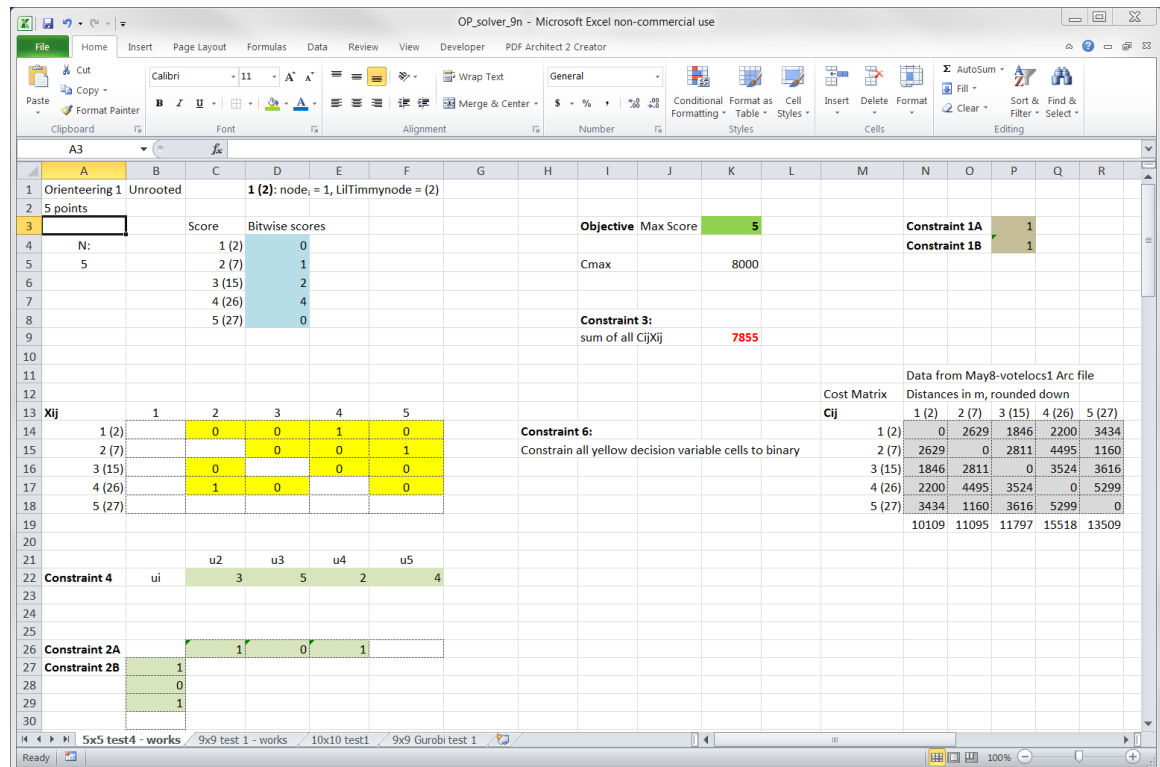
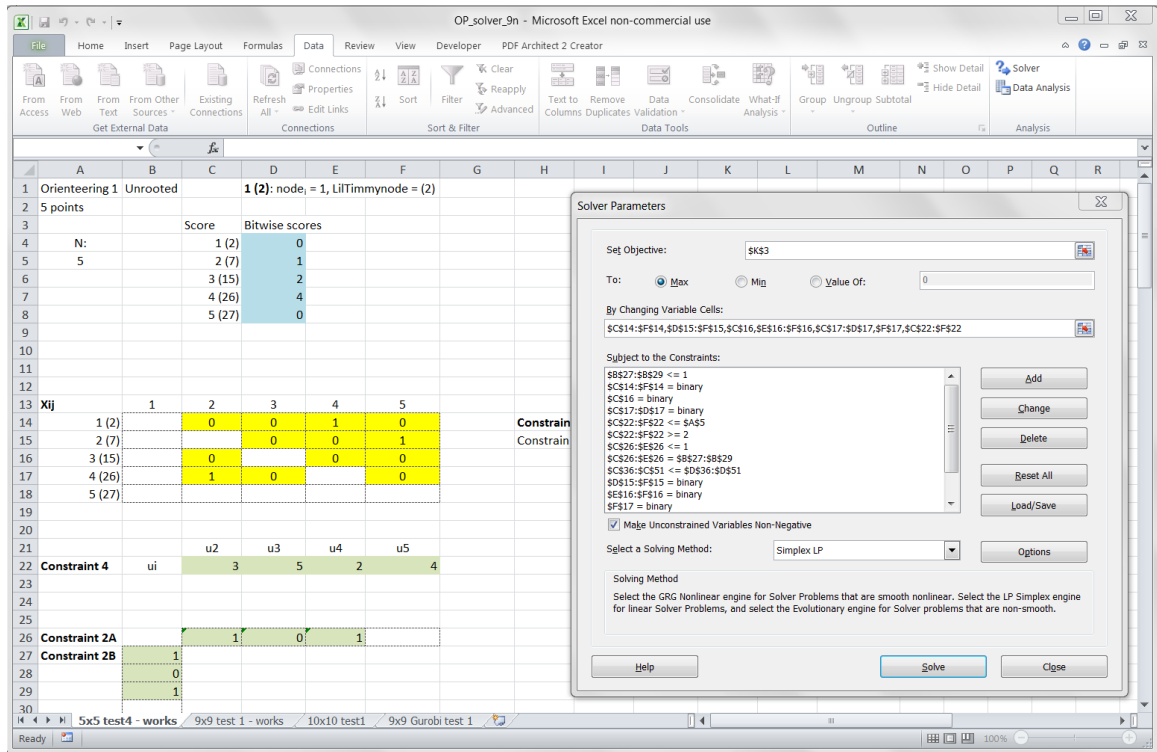


Figure 1: Screenshot of the Excel model for a 5-vertex problem—constraint 5 is built into the sheet in cells below this screenshot.



The next constraint addressed was the first of the two subtour elimination constraints, which works simultaneously in the formulation with the second subtour constraint to assign the vertices to the solution path in an order such that the path would finish at the assigned last vertex (whether the first vertex for the rooted model or the end vertex for the unrooted model). Before the second subtour elimination constraint was constructed correctly to work in conjunction with the first subtour constraint, the “alldifferent” constraint available in the Excel Solver Add-In was utilized in an attempt to ensure that each vertex was assigned to the solution path in a unique position on the path. However, this construction proved not to work in conjunction with the second subtour constraint and the “alldifferent” constraint was removed. The equations of the formulation to be evaluated by the Solver Add-In were entered into a pop-up window, as seen in Figure 2.



**Figure 2: The Excel Solver Add-In pop-up window seen on right showing the objective cell, decision variable cells, and most of the constraints of the model.**

As the second subtour elimination was also under construction, it became clear that to handle that constraint in Excel it was necessary to build a pair of cells to evaluate the constraint for every  $i, j$  pair after the start vertex. After building the second constraint, it was possible to add the cells corresponding to the first subtour constraint to the Excel Solver as decision variable cells with values set to be between 2 and  $n$ , as defined by the formulation. By doing this, the Solver would then assign values ( $u_i$ ) to those cells, and each unique value  $u_i$  assigned would correspond to that vertex's position in the solution path, with the exception of the final vertex. The final vertex was not identified by a unique  $u_i$  value assigned to the first subtour constraint, but would be apparent in the definition of the problem and labeled in the decision variable cell matrix.

A fundamental aspect of the rooted OP formulation that proved challenging to build in Excel was a structure to handle the rooting constraint—that the solution path be rooted and return to the initial vertex on the final arc of the path. As a result of this problem, the decision was made that, while also working on building the other constraints listed above, to switch the model formulation to the unrooted variant of the OP. Since the unrooted OP was the first formulation found in Vansteenwegen et al. (2011) for this research, removing the aspects of the model that were manually added to convert the formulation to a rooted form would likely make building a model in Excel from a formulation that was known to be correct an easier process. This theory proved to be successful for the construction of the first Excel OP model. Similar to the initializing constraint, the sum for the column of all binary decision variable cells where  $j = n$  (except for  $i = n$ ) was set to a total of 1. After switching to the unrooted OP formulation, all of the constraints were added to the Excel model successfully and the model was found to work correctly with five vertices. The five vertex problems were small enough that it was possible to verify that the results were correct with straightforward calculations. Through experimentation, the maximum possible problem size given the Solver constraint and decision variable limits was found to be  $n = 9$ .

### **3.2.2. Gurobi Optimizer**

After the upper bound of tractability was established for the Excel model, work transitioned to development of an optimal solver using an ILP approach. The Gurobi Optimizer was selected to develop an ILP model that accurately represented the OP formulation in order to find the optimal solution for each dataset. The Gurobi Optimizer

is free for educational research use and it is stated on the website that it is “a state-of-the-art solver” that was “designed from the ground up to exploit modern architectures and multi-core processors, using the most advanced implementations of the latest algorithms” (Gurobi Optimization, n.d., paras. 5–6). Rather than constructing a computer program in Python or another language and then calling Gurobi in that program, the first effort to use Gurobi was through writing a specific model in LP format as described in the Gurobi documentation with all relevant problem data (i.e. the distance matrix, etc.) built directly in the model. The main work in creating an LP-format model written in WordPad was to translate the calculations built into the Excel model into accurate formulae using the correct semantics.

The first attempt at constructing a model using LP format was undertaken using a small 5-vertex problem. Converting the Excel model including the constraints constructed in the Excel Solver Add-In was mostly straightforward. The only constraint that needed to be modified from the Excel model structure was subtour elimination constraint 5. To be utilized in the LP format, these formulae were converted into a format that had all variables on the left side of the inequality sign (formula are required to have all variables on the left side of an equality/inequality sign in LP format). The first attempt to build an LP-format model on a 5-vertex problem instance provided expected results that were confirmed against results from the Excel model. The next step was to expand the LP-format model and so a model for a larger 9-vertex problem was then attempted. This effort was ultimately successful as well, though it became clear that the LP-format model itself quickly got much larger with the slightly larger problem. The LP-format

model for the first 5-vertex problem was just 63 lines long; the LP-format problem for the first 9-vertex problem was 222 lines long and a 10-vertex set had a solution model of 298 lines. The 15-vertex model came to 615 lines. The objective function and constraint 1 for a 10-vertex problem written in LP format can be seen in Figure 3; a complete 5-vertex model is included in Appendix C.

```

Maximize
    48 x0202 + 48 x0203 + 48 x0204 + 48 x0205 + 48 x0206 + 48 x0207
+ 48 x0208 + 48 x0209 + 48 x0210 + 77 x0302 + 77 x0303 + 77 x0304
+ 77 x0305 + 77 x0306 + 77 x0307 + 77 x0308 + 77 x0309 + 77 x0310
+ 89 x0402 + 89 x0403 + 89 x0404 + 89 x0405 + 89 x0406 + 89 x0407
+ 89 x0408 + 89 x0409 + 89 x0410 + 12 x0502 + 12 x0503 + 12 x0504
+ 12 x0505 + 12 x0506 + 12 x0507 + 12 x0508 + 12 x0509 + 12 x0510
+ 23 x0602 + 23 x0603 + 23 x0604 + 23 x0605 + 23 x0607 + 23 x0608
+ 23 x0609 + 23 x0610 + 32 x0702 + 32 x0703 + 32 x0704 + 32 x0705
+ 32 x0706 + 32 x0707 + 32 x0708 + 32 x0709 + 32 x0710 + 7 x0802
+ 7 x0803 + 7 x0804 + 7 x0805 + 7 x0806 + 7 x0807 + 7 x0808 + 7
x0809 + 7 x0810 + 39 x0902 + 39 x0903 + 39 x0904 + 39 x0905 + 39
x0906 + 39 x0907 + 39 x0908 + 39 x0909 + 39 x0910

Subject to
\ Constraint 1
    c1A: x0102 + x0103 + x0104 + x0105 + x0106 + x0107 + x0108 +
x0109 + x0110 = 1
    c1B: x0110 + x0210 + x0310 + x0410 + x0510 + x0610 + x0710 +
x0810 + x0910 = 1

```

**Figure 3: First part of a 10-vertex LP-format model run by the Gurobi Optimizer at the command line.**

Three 5-vertex sets, five 10-vertex sets, and five 15-vertex sets were evaluated for optimal solutions across a range of total cost constraint  $C_{max}$  values. Since the unrooted OP variant ensures that the start and end vertices have no bearing on the value of the objective function, this meant that for 5-vertex sets only three of the five vertices impacted the outcome. After limited testing, it was determined that while the 5-vertex

sets were useful for construction and testing of the optimal solution procedures, they were very possibly would be too small for worthwhile evaluation of the heuristics.

The process of constructing models written in LP format was not very technically challenging once the basic conventions were understood. The process was substantially reliant on correct manual data entry. For the purpose of aiding that manual process, values transcribed from ArcGIS were always rounded down to the nearest meter. While this is unlikely to cause significant issues for problems at the scale of a city road network, nevertheless in network space inconsistent measurements could lead to an incorrect result at the margins.

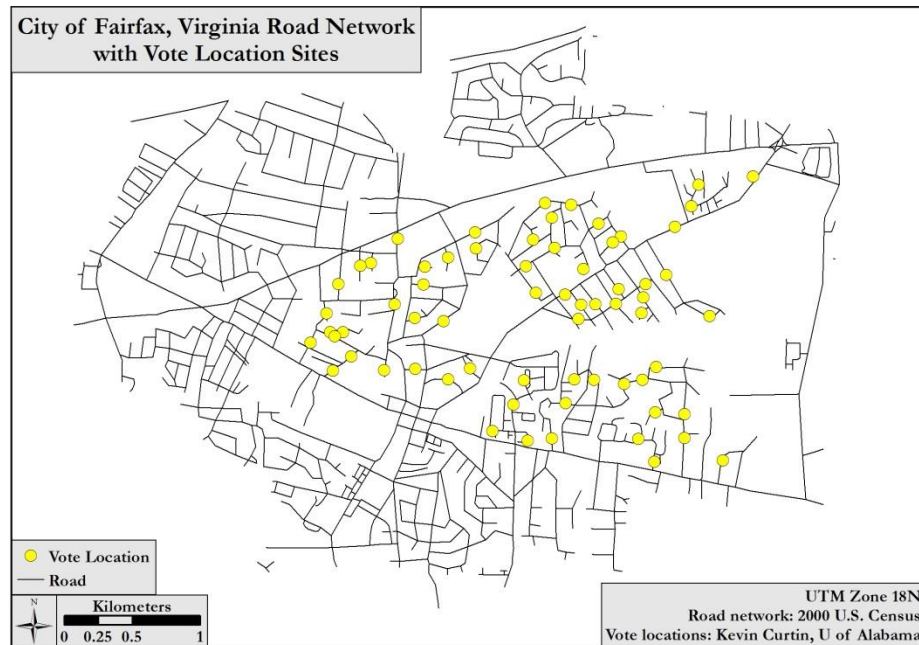
#### 4. DATA

In order to create a small network of data points on which to test various heuristic processes while trying to retain short processing times, road network data and a set of addresses for a small jurisdiction was obtained. A TIGER/line feature from the 2000 United States census for the road network for Fairfax City, Virginia was downloaded from ESRI.com (this data is no longer available at that website). The road network is assumed to be barrier-free for this study.

A list of 66 mailing addresses located in Fairfax city was provided by Dr. Kevin Curtin. Since the jurisdiction is of a small land area—6.24 square miles in 2010 (“U.S. Census Bureau QuickFacts,” n.d.)—the data was projected into Universal Transverse Mercator (UTM) zone 18 north. This projection is designed for use in a region of the surface of the earth that entirely includes Fairfax city, and minimizes distance distortion to four parts in 10,000 along the central meridian of the zone. See Figure 4 for the Fairfax City road network and vote location sites. Using the road network for a small city like Fairfax, Virginia is an appropriate setting for this research because it is a location where many real-life applications of the OP might be used in a day. It is also not a network that to the eye is obviously very irregular or unusual in relation to a typical American town. These attributes render the network appropriate to evaluate heuristics to approach the OP.

To identify appropriate distances to test as the total cost constraint for the study during the initial heuristic development using the rooted OP variant, the total distance to traverse the road network and visit all 66 locations and return to the set root location was

analyzed for all 66 locations using the Find Route tool in ArcGIS. The total distance to travel to all 66 locations and return to the root location ranged from 29,002 m to 30,237 m, and the mean of all 66 minimum distances was 29,158 m. While it is not assumed that these results are all guaranteed optimal due to Curtin et al. (2014), knowing the average minimum total distance to travel to all 66 points provided a useful frame of reference on which to define the total cost constraint later.



**Figure 4: The complete city of Fairfax, Virginia road network and 66 vote location sites.**

Two sets of the 66 address locations were created with random vote totals during initial development and testing of the heuristics. For both datasets, each of the address locations was assigned a random weighting value between 1 and 100 inclusive using the



Random Integer Set Generator tool available from [www.random.org](http://www.random.org). To determine the root location for each set of locations, the order of the set of locations was randomized using the Random Sequence Generator available from [www.random.org](http://www.random.org), and the first location from that process was deemed the root point and assigned a vote total of zero. The sum total of all random vote amounts assigned to these datasets used with the rooted OP after setting the root location to zero was 3,386 for set 1 and 3,207 for set 2.

For the purposes of creating smaller datasets (of 5, 10 , and 15 points) during the development of the optimal solution procedures using the unrooted OP variant, subsets of the 66 Fairfax city points were built. The Random Integer Set Generator on [www.random.org](http://www.random.org) was used to select which of the 66 points to include in 5, 10, and 15-vertex sets and to assign values to the selected vertices. Five sets each of 5, 10, and 15 vertices were created in this fashion.

An important step in creating the datasets on which to evaluate the heuristics was to define an appropriate total cost constraint ( $C_{max}$ ) value. If the  $C_{max}$  is set too high, the OP becomes the TSP because all vertices can be visited. If the  $C_{max}$  is set too low, the solution becomes too straightforward and not an interesting problem to solve. Guidance for setting worthy  $C_{max}$  values was found in Vansteenwegen (as cited in Vansteenwegen et al., 2011), where his research showed that the most difficult OP problems to solve are those where the number of vertices in the solution are just over half the total number of vertices. For the purposes of the research, it was a logical choice to seek the most difficult versions of the problem on which to evaluate the heuristics. As a result, for each 10-vertex dataset the smallest  $C_{max}$  value that solved in six vertices was chosen. Later the

largest  $C_{max}$  value that solved optimally in six vertices was also used to test the heuristics in order to add an additional data point of analysis for each 10-vertex dataset. Using this justification, eight vertices were chosen for the same purpose on the 15-vertex datasets.

The datasets each were solved first to find the smallest  $C_{max}$  that made it possible to visit all 10 vertices in the set—in other words the non-capacitated TSP. Then each set was solved for iteratively smaller  $C_{max}$  values in order to find the smallest  $C_{max}$  value that solved optimally in six or eight vertices respectively. Determining the length of the solution path obtained from implementing a specific model in Gurobi was achieved by inputting the decision variables from the solution into a custom Excel model built for each dataset. The results of this iterative process for each 10 and 15-vertex set can be found in Appendix D.

## 5. RESULTS

Results were generated first using the initial rooted OP formulation applied to the full set of 66 vote locations on the city of Fairfax, VA road network. These initial results can provide some idea of how the heuristics might perform on a medium to large set of points; although without optimal results to compare against the value of these initial results were somewhat limited. After the optimal solution procedures were developed, more results were generated using the unrooted OP formulation applied to smaller subsets of the complete set of 66 vote locations on the same road network. The unrooted OP heuristic results were then generated and evaluated relative to the optimal solutions for each dataset.

### **5.1. Initial Rooted OP Heuristics**

The first results generated for the study came from the initial effort to develop potential solution heuristics for the rooted OP. The NNH and HAH were developed first and applied to the complete set of 66 vote locations geocoded to the city of Fairfax, Virginia road network. Soon after, an initialization step was developed and subsequently tested with the NNH and HAH. The exact definitions for the rooted OP heuristics can be found in Appendix A. Optimal solutions were not obtained for these initial heuristic results to be compared against.

#### **5.1.1. Initial Heuristic Results**

The NNH succeeded in providing solutions for the rooted OP as presented above using the data from Fairfax city, Virginia. For set 1, the heuristic found a solution trip

that would accumulate 1,390 votes, which was 41.1% of the 3,386 total possible votes. The trip had 28 stops and a total length of 14,359 m, less than the 15,000 m total cost constraint, but with 641 m of allowed distance unused. The data of the steps taken using this heuristic on set 1 can be seen in Table B1 in Appendix B, and the path across the road network can be seen in Figure 5. For set 2 the NNH was less successful, accumulating 1,229 votes and therefore 38.3% of the 3,207 possible votes for the dataset. The total trip for set 2 had 25 stops and was 12,944 m, leaving 2,056 m of the cost constraint unused in that case.

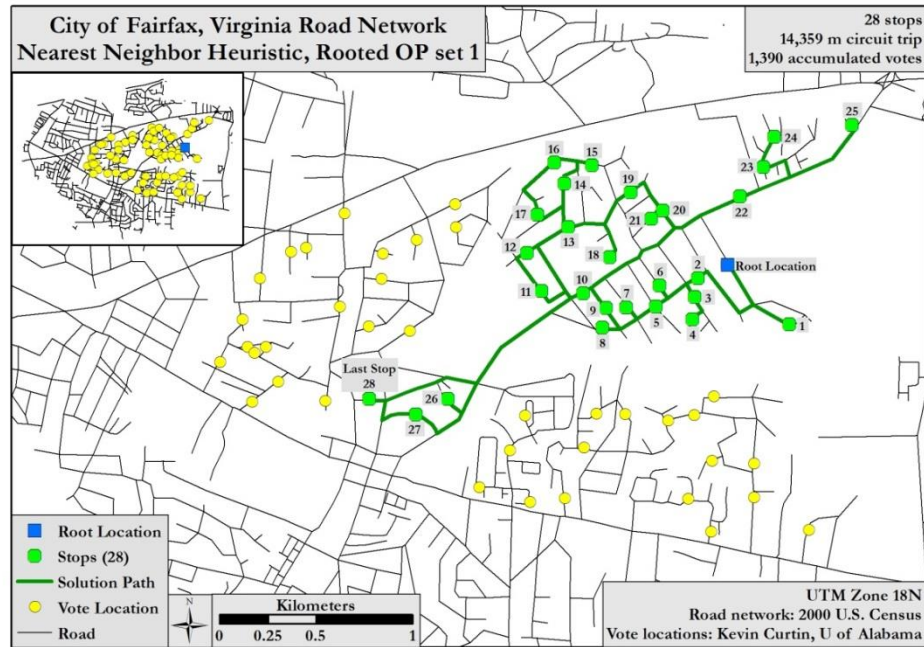
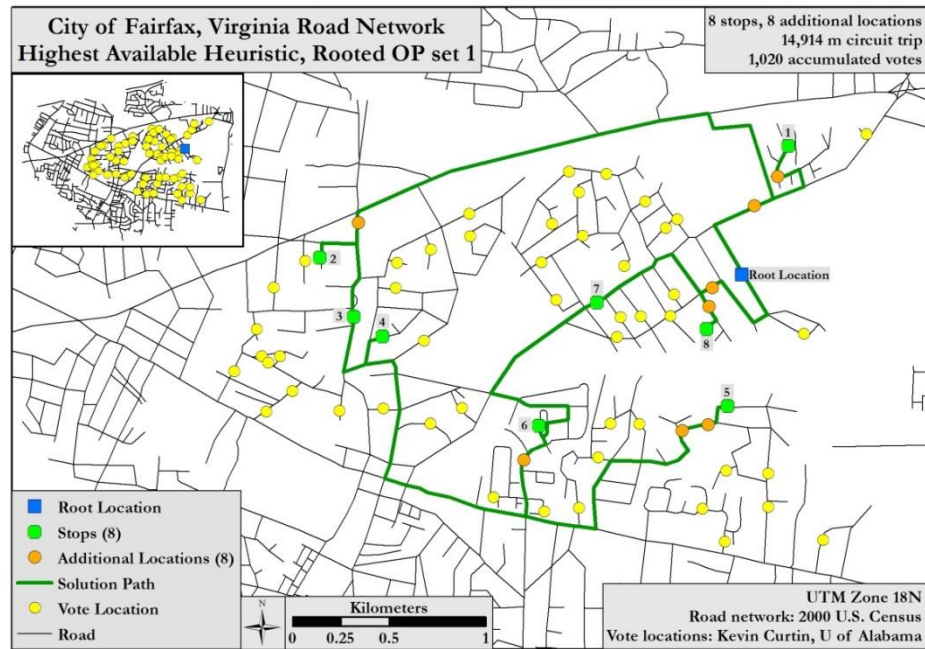


Figure 5: Solution path found by the Nearest Neighbor Heuristic for rooted OP set 1.

The HAH also provided a solution for the rooted OP as described above. For set 1 the heuristic provided a trip with 8 destination stops and 8 additional locations on the way

to those destinations that accumulated 1,020 votes, which was 30.1% of the 3,386 total possible votes. This trip had a total length of 14,914 m, which is just under the total cost constraint of 15,000 m. The results of the steps taken using this heuristic on this set of data can be seen in Table B2 in Appendix B, and the solution path across the road network can be seen in Figure 6. For dataset 2 the HAH found a trip of 9 destination stops and 9 additional locations on the way to those destinations that gathered 1,079 votes, which is 33.6% of the 3,207 possible votes. The trip had a total length of 14,678 m.



**Figure 6: Solution path found by the Highest Available Heuristic for rooted OP set 1.**

### **5.1.2. Initial Heuristics with Initialization Results**

After implementing the NNH and HAH on 2 66-vertex sets of the Fairfax data, the two heuristics were also applied to the same Fairfax datasets with the centralizing initialization step included. The initialization step improved the results found with the NNH. For set 1, the NNH with initialization found a solution trip that accumulated 1,975 votes, which was 58.3% of the 3,386 possible votes. This trip had 33 stops and 1 additional location visited on the way to the set's 1-median, with a total length of 14,943 m. The NNH with initialization gathered 585 more votes for set 1 than the NNH alone, which was a 42.1% increase in votes. For set 2, the NNH with initialization generated a trip that gathered 1,611 votes, which was 50.2% of the 3,207 possible votes. The solution trip for set 2 also had 33 stops and 1 additional location on the way to the set median and was 14,906 m long. The NNH with initialization accumulated 382 more votes for set 2 than the NNH alone, which was a 31.1% increase in votes.

The results for the HAH with initialization were mixed. For set 1, the solution trip that was found accumulated 1,209 votes, which was 35.7% of the possible votes. This trip had 10 stops and 7 additional locations and a total distance of 14,982 m. The HAH with initialization accumulated 189 more votes than the HAH alone for set 1, which was an 18.5% increase in votes. For set 2, the HAH with initialization generated a solution trip that obtained 1,006 votes, which was 31.4% of the total votes possible for this set. The trip had 9 destination stops and 7 additional locations and was 14,856 m long. The HAH with initialization accumulated 73 fewer votes for set 2 than the HAH alone, which was a 6.8% decrease in votes.

### **5.1.3. Initial Heuristic Results Summarized**

The NNH was the stronger performing of the two heuristics developed when applied to the two 66-vertex datasets with the rooted OP by accumulating 41.1% and 38.3% of the total votes possible. These NNH solution paths traveled 49.2% and 44.4% respectively of the mean minimum path to visit all locations. While the HAH is a greedy heuristic—a concept employed in other studies in the OP literature—the HAH accumulated 30.1% and 33.6% of the total votes possible for both datasets on paths that traveled 51.2% and 50.3% of the mean minimum path to visit all locations.

For the address locations and underlying Fairfax city network dataset used with the rooted OP, in three of four cases the heuristic procedures achieved double-digit percentage improvements in votes accumulated when employing the centralizing initialization step. The initialization step was more successful on the NNH, which is a logical result since the solution path is built by adding iteratively the next-closest location to the path. More results would be needed to confirm these preliminary results. All results of the initial heuristic implementations to the rooted OP are displayed in Table 1.

**Table 1: Heuristic Results on the Rooted OP Using Sets of All 66 Vote Locations and  $C_{max}$  of 15,000 m**

Heuristic	Votes	Percent of total	$\Delta$ Votes wI <sup>a</sup>	$\Delta$ Percent votes wI <sup>a</sup>	Stops	No. of add'l locations	Path dist. (m)
Dataset 1							
NNH	1,390	41.1%	-	-	28	-	14,359
NNHwI <sup>a</sup>	1,975	58.3%	+585	+42.1%	33	1	14,943
HAH	1,020	30.1%	-	-	8	8	14,914
HAHwI <sup>a</sup>	1,209	35.7%	+189	+18.5%	10	7	14,982
Dataset 2							
NNH	1,229	38.3%	-	-	25	-	12,944
NNHwI <sup>a</sup>	1,611	50.2%	+382	+31.1%	33	1	14,906
HAH	1,079	33.6%	-	-	9	9	14,678
HAHwI <sup>a</sup>	1,006	31.4%	-73	-6.8%	9	7	14,856

<sup>a</sup> with Initialization

## **5.2. Unrooted OP Optimal Solutions**

Three 5-vertex problem instances were evaluated for optimal solutions by implementing the Gurobi procedure using several different total cost constraint  $C_{max}$  values. After considering that two of the five vertices in each set were predefined with scores of 0, leaving only three vertices at most to be part of a solution, it was decided that 5-vertex problem instances were likely too small to provide a worthwhile testing scenario on which to evaluate the heuristics. The particular details of the dataset (distance values and vertex scores) were likely to play a strong role in the result, possibly more so than the heuristic used in a given circumstance. No data were gathered to validate this hypothesis.

Five sets of 10-vertex problems were then evaluated for optimal solutions using the Gurobi procedure. As described in the Data section, each set was repeatedly solved using iteratively smaller  $C_{max}$  values with an eye towards identifying the largest and smallest  $C_{max}$  values that could be solved optimally with six vertices. For each 10-vertex



set, these  $C_{max}$  values were selected as the values at which to evaluate the heuristics. This progression of results for one dataset can be seen in Table 2, where the first row represents a TSP result for the problem instance since the  $C_{max}$  was set large enough to allow the solution path to include all vertices. For each 15-vertex set,  $C_{max}$  values were sought for the largest and smallest eight vertex optimal solutions in order to search for solutions using just over half the points in the dataset. Tables for the progression of results for all 10 and 15-vertex datasets are included in Appendix D.

**Table 2: Optimal Results for Dataset 10n\_set2 with Progressively Smaller  $C_{max}$  Values**

Dataset & $C_{max}$	Total votes	Total solution distance cost (m)	No. of vertices in solution
10n_set2 10000	372	9,608	10
10n_set2 9607	365	9,557	9
10n_set2 9556	365	9,484	9
10n_set2 9483	352	9,260	9
10n_set2 9259	345	9,137	8
10n_set2 9136	306	9,119	8
10n_set2 9118	305	9,102	8
10n_set2 9101	299	8,372	7
10n_set2 8371	299	8,350	7
10n_set2 8349	299	8,349	7
10n_set2 8348	279	8,002	6
10n_set2 8001	239	7,967	6
10n_set2 7966	234	6,751	8
10n_set2 6750	234	5,754	8
10n_set2 5753	234	5,202	8
10n_set2 5201	227	5,150	7
10n_set2 5149	227	5,078	7
10n_set2 5077	214	4,854	7
10n_set2 4853	207	4,731	6
10n_set2 4730	168	4,713	6
10n_set2 4712	167	4,696	6
10n_set2 4695	161	3,966	5

*Note.* The yellow cell is the smallest  $C_{max}$  to include all 10 vertices in the solution. The green cell is the largest  $C_{max}$  to use 6 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 6 vertices in the solution.

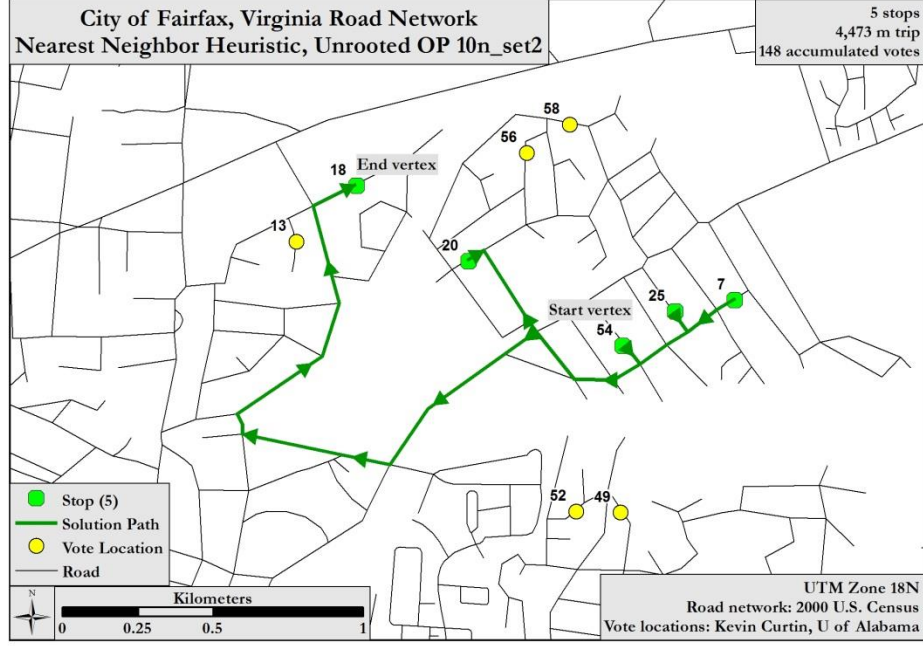
### 5.3. Example Implementation of Unrooted OP Heuristics

The modified heuristics using the unrooted OP formulation were first tested on a 10-vertex set of points (10n\_set2) that were selected at random from the original 66-point Fairfax City dataset. This set of 10 vertices was used with a total cost constraint of 4,696 m, as chosen by the iterative process highlighted in the previous section and shown in Table 2. The NNH found a solution trip that would accumulate 148 votes, which was 39.8% of the 372 total votes that would be gathered from visiting all 10 locations. The trip had 3 stops that accumulated votes (in addition to the start and end vertices) and a total length of 4,473 m, and thus only 223 m of allowed distance unused. The data for the steps taken using this heuristic on 10n\_set2 can be seen in Table 3 and the solution path is displayed in map format in Figure 7.

Table 3: Results of Implementation of NNH on Dataset 10n\_set2 with  $C_{max}$  of 4,696 m.

Step	Origin vertex	Dest. vertex	Votes	Trip dist.	Path subtotal	Dist. to end	Path dist.	Total votes
1	54	25	81	351	351	2,650	3,001	81
2	25	7	60	269	620	2,998	3,618	141
3_1	7	58	43	951	1,571	3,232	4,803	
3_2	7	56	23	1,191	1,811	2,992	4,803	
3_3	7	20	7	1,203	1,823	2,650	4,473	148
4_1	20	56	23	465	2,288	2,992	5,280	
4_2	20	58	43	705	2,528	3,232	5,760	
4_3	20	13	20	2,664	4,487	334	4,821	
4_4	20	49	65	3,318	5,141		5,141	
4_5	20	52	73	3,320	5,143		5,143	
Final	20	18	0	2,650	4,473			148

*Note.* Distances are in meters. Distance subtotals in green were less than the  $C_{max}$  value and the corresponding destination vertices thus were added to the solution path. Subtotals in red violated the  $C_{max}$  constraint and the corresponding destination vertices were discarded for that step.



**Figure 7: Solution path found by the Nearest Neighbor Heuristic for unrooted OP dataset 10n\_set2 with  $C_{max}$  of 4,696m.**

The HAH also provided a solution for the unrooted OP. For dataset 10n\_set2 the heuristic provided a solution path with 3 stops that accumulated 161 votes (with no additional locations on the way to those stops), which was 43.3% of the 372 total votes that would be gathered from visiting all 10 locations. This trip had a total length of 3,966 m, which left 730 m unused under the total cost constraint of 4,696 m. The data of the steps taken using this heuristic on this set of data can be seen in Table 4 and the solution path is displayed in map format in Figure 8.

Table 4: Results of Implementation of HAH on Dataset 10n\_set2 with  $C_{max}$  of 4,696 m.

Step	Origin vertex	Dest. vertex	Votes	Trip dist.	Path subtotal	Dist. to end	Path dist.	Total votes
1	54	25	81	351	351	2,863	3,214	81
2_1	25	52	73	3,533	3,884	3,523	7,407	
2_2	25	49	65	3,530	3,881	3,520	7,401	
2_3	25	7	60	269	620	2,998	3,618	141
3_1	7	52	73	3,669	4,289	3,523	7,812	
3_2	7	49	65	3,666	4,286	3,520	7,806	
3_3	7	58	43	951	1,571	3,232	4,803	
3_4	7	56	23	1,191	1,811	2,992	4,803	
3_5	7	13	20	3,012	3,632	334	3,966	161
4_1	13	52	73	3,536	7,168			
4_2	13	49	65	3,534	7,166			
4_3	13	58	43	3,246	6,878			
4_4	13	56	23	3,005	6,637			
4_5	13	20	7	2,664	6,296			
Final	13	18	0	334	3,966			161

Note. Distances are in meters. Distance subtotals in green were less than the  $C_{max}$  value and the corresponding destination vertices thus were added to the solution path. Subtotals in red violated the  $C_{max}$  constraint and the corresponding destination vertices were discarded for that step.

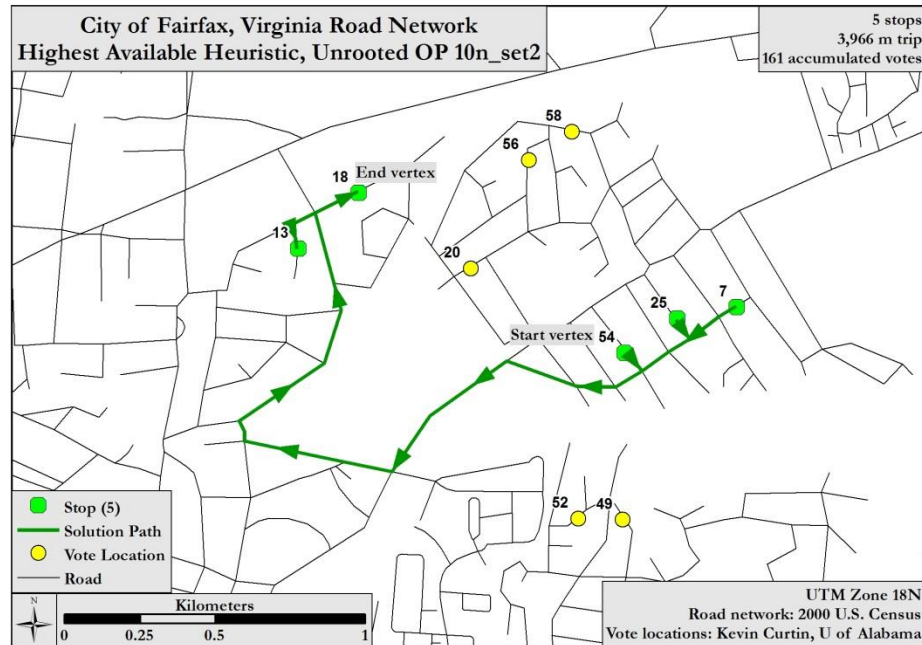


Figure 8: Solution path found by the Highest Available Heuristic for unrooted OP dataset 10n\_set2 with  $C_{max}$  of 4,696m.

#### **5.4. Unrooted OP Results: Heuristics Versus Optimal**

The NNH and HAH results for each dataset evaluated for the unrooted OP can be compared against the optimal solution found using the Gurobi approach. For the first dataset itemized in the previous section, the solver found that with a total cost constraint  $C_{max}$  of 4,696 m, the optimal solution for an unrooted OP applied to 10n\_set2 travelled to 4 stops that accumulated 167 votes, which was 44.9% of the 372 total possible votes. Therefore, the NNH gathered 88.6% of the optimal solution, and the HAH gathered 96.4% of the optimal solution in that case. After completing the first set of 10 vertices, four additional sets of 10 vertices were evaluated with the heuristics. These sets were evaluated at two different  $C_{max}$  values—both the largest and the smallest value that resulted in an optimal result that used six vertices (just over half the vertices). After evaluating five sets of 10 vertices at two different  $C_{max}$  values, five sets of 15 vertices were evaluated in the same manner. The heuristic performance results compared to the optimal solution for each dataset and  $C_{max}$  value are summarized in Table 5.

The combination of five sets each of 10 and 15 vertices with two  $C_{max}$  values evaluated for each set provided 19 results for the NNH and HAH (one dataset only had one  $C_{max}$  value that solved in the desired number of vertices). Due to three of the ten datasets having the start vertex coincide with the 1-median of the dataset, only 13 of the dataset and  $C_{max}$  combinations made it possible to evaluate heuristics with the initialization step. Across the 19 dataset and  $C_{max}$  combinations, the NNH provided results that averaged to 79.7% of the optimal solution for each set. For the 13 sets where the start vertex was not already the 1-median of the set, the NNHwI performed worse

than the NNH alone by averaging 70.7% of the optimal solution. The HAH achieved an average result slightly better than the NNH with 81.6% of the optimal solution. The HAHwI yielded an average 80.7% of the optimal solution for the applicable sets.

**Table 5: Heuristic Performance Relative to Optimal Results for the Unrooted OP**

<b>Dataset &amp; <math>C_{max}</math> (m)</b>	<b>Optimal votes</b>	<b>NNH votes</b>	<b>Percent optimal</b>	<b>HAH votes</b>	<b>Percent optimal</b>	<b>NNHwI<sup>a</sup> votes</b>	<b>Percent optimal</b>	<b>HAHwI<sup>a</sup> votes</b>	<b>Percent optimal</b>
10n_1 4,961	169	169	100%	169	100%	162	95.9%	162	95.9%
10n_1 6,822	271	169	62.4%	271	100%	169	62.4%	271	100%
10n_2 4,696	167	148	88.6%	161	96.4%	c	c	c	c
10n_2 7,616	269	234	87.0%	154	57.2%	c	c	c	c
10n_3 5,031 <sup>b</sup>	219	156	71.2%	188	85.8%	120	54.8%	120	54.8%
10n_4 3,843	246	246	100%	197	80.1%	197	80.1%	197	80.1%
10n_4 7,487	288	246	85.4%	288	100%	246	85.4%	288	100%
10n_5 2,339	151	151	100%	139	92.1%	c	c	c	c
10n_5 7,502	237	183	77.2%	205	86.5%	c	c	c	c
15n_1 8,426	352	224	63.6%	278	79.0%	222	63.1%	278	79.0%
15n_1 10,373	423	277	65.5%	341	80.6%	366	86.5%	315	74.5%
15n_2 7,898	386	347	89.9%	291	75.4%	c	c	c	c
15n_2 8,825	481	431	89.6%	299	62.2%	c	c	c	c
15n_3 6,984	224	224	100%	185	82.6%	174	77.7%	185	82.6%
15n_3 8,761	393	205	52.2%	322	81.9%	224	57.0%	326	83.0%
15n_4 8353	341	255	74.8%	193	56.6%	140	41.1%	193	56.6%
15n_4 9122	341	255	74.8%	208	61.0%	193	56.6%	208	61.0%
15n_5 5029	301	186	61.8%	268	89.0%	269	89.4%	271	90.0%
15n_5 8754	431	301	69.8%	361	83.8%	301	69.8%	397	92.1%

<sup>a</sup> with Initialization

<sup>b</sup> 10n\_set3 only had one optimal solution that used 6 vertices across all  $C_{max}$  values

<sup>c</sup> Initialization not applied because start vertex was already the 1-median of the dataset

Descriptive statistics for the percent optimal results were calculated for each heuristic and heuristic with initialization evaluated in the study. Since the best fitting distribution for the particular phenomena observed in this study was not known, the student's t-test distribution was selected to generate statistics because it has fatter tails than the normal distribution. Therefore the student's t-test can accommodate some of the variance that comes from sampling and that the complete population of datasets on which

the heuristics could be tested is not known. These results are presented in Table 6. It is important to remember that these descriptive statistics just apply to the implementation of these heuristics in the conditions described in the study, including the number of vertices in the datasets and the underlying network. These summary statistics might change substantially if the heuristics were to be systematically applied to datasets of more points on the same underlying network or if other characteristics were to be changed, including the problem formulation or range of scores (votes) applied to vertices in the set.

**Table 6: Means, Standard Deviations, and 95% Confidence Intervals for Percentage of Optimal Measurements for Each Heuristic**

Heuristic	<i>n</i>	<i>M</i>	<i>SD</i>	95% <i>CI</i>	
				<i>LL</i>	<i>UL</i>
NNH	19	79.7%	15.1%	72.4%	86.9%
NNHwI	13	70.7%	16.4%	60.8%	80.7%
HAH	19	81.6%	14.0%	74.9%	88.3%
HAHwI	13	80.7%	15.5%	71.4%	90.1%

Separating out the results for each heuristic by the number of vertices in the test datasets provided a clear distinction in performance for all heuristics. All four heuristics and heuristics with initialization performed better on average on the smaller 10-vertex sets than on the 15-vertex sets. Summary descriptive statistics for this breakdown can be seen in Table 7.

**Table 7: Means, Standard Deviations, and 95% Confidence Intervals for Percentage of Optimal Measurements for Each Heuristic, by Dataset Size**

Heuristic	No. of vertices	$n$	$M$	$SD$	95% $CI$	
					$LL$	$UL$
NNH	10	9	85.6%	13.5%	75.4%	96.1%
NNH	15	10	74.2%	14.9%	63.5%	84.9%
NNHwI	10	5	75.7%	16.8%	54.8%	96.6%
NNHwI	15	7	67.6%	16.5%	53.9%	81.4%
HAH	10	9	88.9%	13.8%	78.1%	99.3%
HAH	15	10	75.2%	11.2%	67.2%	83.2%
HAHwI	10	5	86.1%	19.3%	62.1%	110.1%
HAHwI	15	7	77.3%	12.8%	66.6%	88.0%

No absolute measurements of the effectiveness of the heuristics were determined for this study. However, relative measurements of the heuristics can provide information on performance beyond the summary descriptive statistics already presented. Paired t-tests were performed for each heuristic and heuristic with initialization combination to determine if there was a statistically significant difference between the percent optimal measurements for each heuristic and all other heuristics. These results, using  $\alpha = 0.05$ , can be found in Table 8. Examining the one-tail p-values resulting from this analysis indicated that it was not possible to reject the null hypothesis that there was no difference in the percent optimal performance of the heuristics in four of six pairings. However, two pairings did show a significant difference: HAHwI – NNHwI with a one-tail  $p = 0.0103$  and HAH – NNHwI with a one-tail  $p = 0.0024$ . Drilling down further, the HAHwI – NNHwI pairing showed that the HAHwI had a significantly larger mean percent optimal measurement with a two-tail  $p = 0.0205$ . In addition, the HAH – NNHwI pairing showed that the HAH had a significantly larger mean percent optimal measurement with a two-



tail  $p = 0.0048$ . The statistical summary for the two pairings where a statistically significant difference was found can be found in Table 9.

**Table 8: Statistical Summary of Difference in Percentage Optimal Measurements between Heuristics**

Heuristics compared	n	<i>M</i>	<i>SD</i>	$t_{\text{obs}}$	p (one-tail)	95% CI	
						LL	UL
HAH - NNH	19	1.9%	20.2%	0.4133	0.3421	-7.8%	11.6%
NNH - NNHwI	13	4.8%	17.3%	0.9927	0.1702	-5.7%	15.2%
HAH - HAHwI	13	2.4%	9.2%	0.9331	0.1846	-3.2%	7.9%
HAHwI - NNHwI	13	10.0%	13.5%	2.6673	0.0103	1.8%	18.1%
HAHwI - NNH	13	5.2%	21.1%	0.8938	0.1945	-7.5%	18.0%
HAH - NNHwI	13	12.4%	12.9%	3.4452	0.0024	4.5%	20.2%

**Table 9: Statistical Summary of Difference in Percentage Optimal Measurements for Heuristics Found Significantly Different**

Heuristics compared	n	<i>M</i>	<i>SD</i>	$t_{\text{obs}}$	p (two-tail)	95% CI	
						LL	UL
HAHwI - NNHwI	13	10.0%	13.5%	2.6673	0.0205	1.8%	18.1%
HAH - NNHwI	13	12.4%	12.9%	3.4452	0.0048	4.5%	20.2%

## 6. CONCLUSIONS

Two heuristics and an initialization step were developed that provide solutions to the OP using just a GIS software program and a spreadsheet. Given that there are numerous real-world applications to the OP in generalized form as well as various specialized forms, making OP solution approaches more widely accessible through heuristics that can be implemented with just GIS software and a spreadsheet is a worthwhile goal that was accomplished in this research.

These heuristics can be applied by GIS professionals or others with a limited training in GIS software and without knowledge of linear programming, Python, or other computer programming languages. After completing the initial setup of building the OD Cost Matrix for the problem set, the remaining work in implementing the heuristics was technically trivial and should be accessible with a basic understanding of ArcGIS. A consideration for users employing these heuristics with just a GIS software package and a spreadsheet may be that complete application of the heuristics can be time and labor-intensive with large data sets. However, the heuristics in the study are fairly simple concepts and could serve as the basis for more complex or customized models for a variety of problem scenarios.

The Nearest Neighbor Heuristic is of a basic design and the performance of the heuristic is very dependent on the data inherent to the particular problem. The distribution of vertices across the network, the scores affiliated with the vertices, and the layout of the network itself could change dramatically the effectiveness of the heuristic and the results

found relative to the optimal solution. Initializing the heuristic with a first step to the 1-median location of the network dataset did not prove to be an effective supplement to the heuristic with the datasets tested in this study—in fact for the 13 sets where the initialization step was implemented the heuristic result averaged a lower percentage of the optimal result than the NNH did itself. More testing is suggested before eliminating the initialization step from consideration completely, especially since the initialization step improved results in both cases when applied to the larger 66-point datasets. The initialization step could be appropriate for problem scenarios with a remote root location or vertex set with a mix of clustered and dispersed vertices. The heuristic could present strong appeal for approaching OP scenarios where the simplest or fastest available answer was needed and optimality was a secondary consideration. In addition, the concepts of the heuristic may itself be appropriate as a subset of a more complex heuristic, such as an initialization phase to find an initial feasible solution that could be improved through some other iterative process.

The greedy Highest Available Heuristic performed slightly better than the NNH in this series of tests on average, though not to a statistically significant degree. This heuristic could prove to be the most effective option given certain data parameters, such as on a set of vertices with a very wide range of scores or a more clustered network, or potentially on a problem scenario with a restrictive total cost constraint. Employing the centralizing initialization step with the heuristic may improve the result found or may prove to be more inefficient, depending on the dataset of the problem. Instances of the

initialization step in some cases improving and other times curtailing performance of the HAH were observed in the study.

In terms of statistically significant conclusions achieved, it can be stated with 95% confidence that for the particular parameters evaluated in this study (unrooted OP, Fairfax City road network, vertex scores ranging from 1 to 100, 10 or 15 vertices per dataset, total cost constraints set to achieve optimal solutions using just more than half the total vertices per dataset), the NNHwI average percent optimal measurement was found to be lower than the HAH and HAHwI percent optimal measurements. Additional evaluation is suggested before concluding that the HAH and HAHwI would perform markedly better than the NNHwI in other situations outside the specific problem parameters of this study.

While the results from testing the heuristics using the rooted OP formulation on the full set of 66 points can only be seen as anecdotal, the initialization step did provide improvement over the original heuristic in three of four cases—substantial improvements of 18.5%, 31.1%, and 42.1%. Again, this is not a statistically significant result or large enough sampling upon which to draw any conclusions, but these particular results may indicate that an initialization step is more beneficial on larger datasets when compared against the 10 and 15-vertex datasets where the initialization step led to reduced performance. Such reduced performance from using initialization was observed relative to the original heuristic both for 10 and 15-vertex sets together and for the subsets of just 10-vertex sets and 15-vertex sets separately.

Objective evaluation of whether any heuristic performed sufficiently well to recommend implementation was not achieved in the study. What standard is sufficient for a user to determine that using one or more of the proposed heuristics would be more beneficial than simply eyeballing a given dataset? Would a user want a heuristic to regularly achieve 90% optimality? 80% optimality? Of course the particular application and user needs play a role here as well. If a user requires a consistently high solution performance—near optimal—then the proposed heuristics may not be sufficient and investing in OR expertise or optimization software or both would be called for. However, for circumstances where access to those resources is limited or nonexistent, the heuristics developed here can provide at the very least a starting point for tackling a real-world OP application.

Testing the heuristics with and without the initialization step on more datasets is needed to get a better sense of the likely performance of each heuristic procedure. It is not difficult to imagine that the nature of test data could dramatically alter the results found in this study. It also may be possible to identify acceptable performance limits for one or more heuristic with further evaluation. Evaluation dataset parameters that should be varied beyond what was done in this study for additional testing include: varied range of scores (votes) applied to vertices; number of vertices in test datasets; amount of capacitation set by the cost constraint, i.e. relatively smaller and larger  $C_{max}$  values than those chosen for this study; nature of the underlying network—Manhattan/rectilinear, radio-concentric, sprawling suburban, etc.; and other types of networks such as American vs. European cities, bicycle paths, or utility networks. For example, if the scores assigned

to the locations varied more than was done here—if the events the politician attended on the tour had from 1 to 1,000 attendees—then the greedy heuristic might perform relatively better and the results could vary significantly.

All of the heuristic procedures presented in this study can be modified for application to rooted variations of the OP, and this should be fairly straightforward in most cases. Examples of such modifications are included in Appendix A. Modification of these heuristics for application for other variations and subclasses of the OP should be possible but may be significantly more involved.

With the development of these first heuristics designed specifically for finding solutions to the OP using just GIS software and a spreadsheet, GIS users now have options available to find solution paths for this subclass of problems. While ArcGIS has a built-in function that finds a solution path for the basic TSP, which may or may not be optimal, real world situations are often more closely simulated through the model of the OP. No complex computer programming or specialized software is required to use these heuristics; however, users should be aware that the heuristics do not guarantee optimal results, and in some cases results may be significantly suboptimal.

## 7. FUTURE RESEARCH OPPORTUNITIES

Several areas of research are suggested by the results of the study. The heuristic approaches presented here should be further tested against different data sets and networks of varying sizes, as well as cost constraints ranging from minimal to nearly the full length of the network. The heuristics should also be evaluated fully against optimal solution procedures using the rooted OP. Any underlying network can be used multiple times by varying the values applied to each vertex (both in larger and smaller ranges than the 1-100 range employed here), adding and removing vertices, and varying the total cost constraint applied to multiple test problems. As for the underlying network, the heuristics should be tested on larger networks such as larger American and European cities to see if they are still viable—not only in terms of being able to find an optimal or near-optimal solution, but also with respect to the feasibility of processing time given that application of the heuristics without programming may become extremely time intensive for the user. The heuristics should also be tested on different types of underlying networks, from rectilinear or Manhattan networks to more varied and disparate suburban or rural networks. Testing the heuristics on classic datasets from the TSP and OP literature should also be pursued where possible. More testing of the heuristics on a wider range of problems will provide more credibility to the conclusions reached about the relative value of each heuristic approach.

Additional heuristic concepts should be tested and compared against those proposed here, such as a heuristic that might be considered a “maximum location”

heuristic, which would be designed to generate solutions that include as many destination points as possible. Heuristics employing a combination of concepts may also be successful, such as a heuristic that starts with a greedy approach like the HAH and then switches over to a nearest neighbor concept partway through. Such an approach may offer solid performance towards the objective function goal with the benefit of improved implementation times. Additional techniques might be tested including other initialization steps, backwards-looking approaches as suggested by research into experienced orienteering competitors, and the utilization of cluster analyses to identify potential locations to add to the solution path. Tools like network point cluster analysis available through the SANET program—which works as an extension to ArcGIS—may provide for the development of more complex heuristics that can still be implemented in a GIS setting without complex LP programming (Okabe & Okunuki, 2013). Research presented by Mei (2015) could also suggest methods to identify clusters of points or areas of interest in network space that could be used in future heuristics. Work should also be conducted to better understand the data conditions under which the initialization step presented here would be beneficial or harmful to implementation with a heuristic.

Additionally, the heuristics presented here may be improved upon through further refinements such as additional steps or post-processing methods that could be added. And, just as heuristics are needed for the OP as a subclass of the TSP with additional constraints, so too are heuristics needed for other variations and subclasses of the OP such as the OPTW, OP with compulsory vertices, and TD-OP. Modified versions of the



heuristics in this study may be appropriate for some subclass problems of the OP, and new heuristic approaches may prove best for other problem types.

A primary area of further research should be examining methods and techniques to improve the time and effort required to implement these and future heuristics implemented only with GIS software and a spreadsheet. While these heuristics that can be applied without complex programming approaches present the desirable qualities of being easier to implement and not requiring additional expensive software packages, they can be time and labor intensive. Opportunities to capitalize on additional capabilities built in to ArcGIS might include utilizing ModelBuilder or Python scripting within the software to automate and speed up some aspects of heuristic implementation. In addition, with some knowledge of traditional programming techniques and widely available languages like Python, it may be possible to implement the heuristics presented here in steps taking place both inside and outside the GIS platform. An added benefit to increased automation in heuristic implementation should be a reduced likelihood of error due to the manual data entry involved in the current processes.

Improving the processing time for existing heuristics could also involve designing more efficient methods to achieve the same results, from more quickly eliminating poor solution options to better initiation steps at the beginning of the heuristic. For example, as the greedy HAH approaches the cumulative cost constraint, at some point (perhaps within some percent of the cost constraint or when half the remaining possible locations are still within the distance constraint) it may be more efficient to search for the nearest still-available locations first and then employ the greedy concept to identify the next point to

add to the solution rather than continuing to iterate through locations that are no longer feasible. One or more of the steps outlined above could dramatically reduce the implementation times experienced and shift tractability concerns strictly to problem size and away from heuristic implementation times.

As improved processing power becomes necessary to find solutions to larger and more complex problem sets, a combination solution approach utilizing LP or ILP to solve the problem with a GIS user interface to make the process accessible to lay users may be the only viable approach available. Alternately, a user interface (UI) might be constructed that would allow the user to solve a given OP using one of the developed heuristics within a software suite such as ArcGIS without requiring knowledge of linear programming. Such an interface may increase the size of the problem that can be solved by one of the developed heuristics by dramatically lowering the time requirements for implementation on the user. An additional factor that may lead to improved processing times might be found in the OP formulations themselves. Research by Palomo-Martínez, Salazar-Aguilar, and Albornoz (2017) suggested that it may be possible to implement subtour elimination constraints that perform better than the standard Miller-Tucker-Zemlin TSP formulation constraints frequently cited and used here.

In order to better understand the performance of the heuristics, additional research efforts should pursue comparing the application of these heuristics using just a GIS program and spreadsheet to other approaches using complex programming. The results achieved and processing time required are both important results that should be quantified and compared along with the costs and expertise needed to apply any

particular approach so that potential users can make an informed decision as to what approach is best for a given application. Furthermore, additional research of the literature may provide a recommended percentage of optimal that a heuristic should be expected to achieve in similar circumstances to the research parameters. Alternately, individual user best-guess solutions could be gathered and compared against heuristic performance for additional context.

## APPENDIX A – ROOTED OP HEURISTICS DEFINITIONS

### Nearest Neighbor Heuristic

A relatively simple heuristic to implement is the Nearest Neighbor Heuristic (NNH). This path construction heuristic starts at the root location, finds the nearest location on the network that is not already in the solution path, and continues to add locations while checking to make sure the path includes room under the total cost function for the trip back to the root to complete the circuit.

Nearest Neighbor Heuristic:

Step 1: Start at the root location  $v_1$ .

Step 2: Find the nearest available vote location.

- a. Using the OD Cost Matrix, find the trip length to the nearest available undiscarded vote location  $v_j$  that is not already on the solution path and check to see if the point is within the total cost constraint  $C_{max}$ .
- b. If there are no unselected and undiscarded destinations available within the total cost constraint  $C_{max}$ , proceed to Step 3.
- c. Ensure that there is sufficient distance remaining in the total cost constraint  $C_{max}$  to return to the root location  $v_1$  from the proposed destination. If there is not sufficient distance remaining, the destination cannot be selected and is discarded for this iteration. Return to Step 2(a).
- d. Compute the distance traveled on the trip.

- e. Add the distance traveled on the trip to the total distance traveled, and the votes obtained on the trip to the total votes.
- f. Proceed from the selected destination to Step 2(a).

Step 3: Once it is shown that no more locations can be added to the trip without violating the total cost constraint, including the distance needed to return to the root location, check the trip from the last destination on the path to the root location for any additional destinations that have not been added to the total trip thus far. Add the vote totals from any locations on the final trip to the root location to the overall total votes accumulated.

Each tested destination was catalogued in a spreadsheet to track all visited and rejected destinations, total votes accrued, and the total distance traveled. To find any additional destinations located on the final path back to the root in Step 3, the Find Route function was used to draw the shortest path between the last point on the path and the root location.

### **Highest Available Heuristic**

The Highest Available Heuristic (HAH) uses many of the basic mechanics of the Nearest Neighbor Heuristic, but is built as a greedy heuristic by searching iteratively for the highest scoring location available.

Highest Available Heuristic:

Step 1: Start at the root location  $v_1$ .

Step 2: Find the highest available vote location.

- a. Select the highest available undiscarded vote location  $v_j$ .

- b. Using the OD Cost Matrix, find the trip length to the highest available undiscarded vote location  $v_j$  and check to see if the point is within the total cost constraint  $C_{max}$ .
- c. If there are no unselected and undiscarded destinations available within the total cost constraint  $C_{max}$ , proceed to Step 3.
- d. Ensure that there is sufficient distance remaining in the total cost constraint  $C_{max}$  to return to the root location  $v_1$  from the proposed destination. If there is not sufficient distance remaining, the destination cannot be selected and is discarded for this iteration. Return to Step 2(a).
- e. Select and note all additional destinations that are located on the trip.
- f. Compute the distance traveled on the trip and total votes obtained at all destinations.
- g. Add the distance traveled on the trip to the total distance traveled, and the votes obtained on the trip to the total votes.
- h. Proceed from the selected destination to Step 2(a).

Step 3: Once it is shown that no more locations can be added to the trip without violating the total cost constraint, including the distance needed to return to the root location, check the trip from the last destination on the path to the root location for any additional destinations that have not been added to the total trip thus far. Add the vote totals from any locations on the final trip to the root location to the overall total votes accumulated.

Again, each tested destination was catalogued in a spreadsheet to track all visited and rejected destinations, total votes accrued, and the total distance traveled. To find all additional destinations located on a tested trip, the Find Route function was used to draw the shortest path between the tested origin and destination points.

## APPENDIX B – ROOTED OP HEURISTICS IMPLEMENTATION OUTPUT

Table B1: Steps taken to apply the Nearest Neighbor Heuristic to set 1 using the rooted OP.

Step	Origin #	Loc #	Votes	Trip Distance	Total Distance	Distance to Root	Circuit Distance	Total Votes
1	1	45	69	459.41	459.41	459.41	918.81	69
2	45	7	54	627.92	1087.33	654.72	1742.05	123
3	7	34	47	149.25	1236.57	803.96	2040.54	170
4	34	42	83	150.72	1387.29	954.68	2341.97	253
5	42	60	9	428.44	1815.73	918.46	2734.18	262
6	60	25	66	151.39	1967.11	876.72	2843.83	328
7	25	54	40	351.87	2318.98	1090.48	3409.47	368
8	54	44	48	275.47	2594.45	1231.28	3825.73	416
9	44	40	62	222.12	2816.57	1119.01	3935.58	478
10	40	4	87	179.17	2995.74	1035.69	4031.42	565
11	4	22	24	282.21	3277.95	1317.90	4595.85	589
12	22	20	10	276.87	3554.82	1477.46	5032.27	599
13	20	6	78	253.02	3807.84	1224.43	5032.27	677
14	6	56	52	264.99	4072.83	1392.85	5465.68	729
15	56	58	41	240.38	4313.21	1152.47	5465.68	770
16	58	31	43	209.70	4522.92	1362.17	5885.09	813
17	31	5	81	371.84	4894.76	1541.80	6436.56	894
18	5	61	12	697.29	5592.05	1230.04	6822.10	906
19	61	19	2	414.74	6006.79	817.89	6824.68	908
20	19	55	80	263.24	6270.03	554.65	6824.68	988
21	55	66	16	88.24	6358.27	642.90	7001.17	1004
22	66	14	64	579.61	6937.88	526.08	7463.96	1068
23	14	27	28	495.13	7433.01	1021.21	8454.22	1096
24	27	46	100	189.43	7622.44	1197.70	8820.13	1196
25	46	12	31	856.03	8478.47	1260.05	9738.51	1227
26	12	41	84	2720.79	11199.26	2050.11	13249.37	1311
27	41	21	8	428.98	11628.24	2269.71	13897.94	1319
28	21	43	71	374.65	12002.89	2355.71	14358.60	1390
Return	43	1	0	2355.71	14358.60			1390

*Note.* Omitted for brevity are 37 steps attempted to find a 29<sup>th</sup> point to add to the solution path. Distances are in meters. Distance subtotals in green were less than the  $C_{max}$  value and the corresponding destination vertices thus were added to the solution path.



Table B2: Steps taken to apply the Highest Available Heuristic to set 1 using the rooted OP.

Step	Origin #	Loc #	Votes	Loc #2	Votes	Loc #3	Votes	Trip Distance	Total Distance	Distance to Root	Circuit Distance	Total Votes
1	1	46	100	27	28	14	64	1197.70	1197.70	1197.70	2395.40	192
2	46	17	99	47	15			3561.04	4758.74	3481.24	8239.99	306
3	17	57	96					648.30	5407.04	2903.02	8310.06	402
4	57	39	93					609.63	6016.67	2684.76	8701.43	495
5	39	16	92	51	11	10	58	3342.16	9358.82	4547.89	13906.71	656
6.1	16	15	91					3204.89	12563.71	3184.65	15748.36	
6.2	16	9	90	64	3			2169.41	11528.24	2514.97	14043.20	749
7.1	9	15	91					2135.12	13663.35	3184.65	16848.00	
7.2	9	53	88					1650.37	13178.60	4028.84	17207.44	
7.3	9	4	87					1479.28	13007.51	1035.69	14043.20	836
8.1	4	15	91					2148.96	15156.47			
8.2	4	53	88					2993.16	16000.67			
8.3	4	41	84					1014.43	14021.94	2050.11	16072.05	
8.4	4	42	83	34	47			951.67	13959.18	954.68	14913.86	966
Return	42	1	0	7	54			954.68	14913.86			1020

*Note.* Distances are in meters. Distance subtotals in green were less than the  $C_{max}$  value and the corresponding destination vertices thus were added to the solution path. Subtotals in red violated the  $C_{max}$  constraint and the corresponding destination vertices were discarded for that step.

## APPENDIX C – SAMPLE OPTIMAL MODEL GUROBI CODE

The following code will find the optimal solution for a specific 5-vertex unrooted OP dataset with a cost constraint  $C_{max}$  set at 10,000 meters. This model was written in WordPad and saved with the file extension .LP. The model was then executed in the Gurobi Interactive Shell with a command at the command prompt such that the values for the objective function and each decision variable were written to a new text file or displayed on screen.

```
Maximize
    22 x22 + 22 x23 + 22 x24 + 22 x25 + 77 x32 + 77 x33 + 77 x34 +
    77 x35 + 94 x42 + 94 x43 + 94 x44 + 94 x45

Subject to
\ Constraint 1
    c1A: x12 + x13 + x14 + x15 = 1
    c1B: x15 + x25 + x35 + x45 = 1

\ Constraint 2
    c2A1: x12 + x22 + x32 + x42 <= 1
    c2A2: x13 + x23 + x33 + x43 <= 1
    c2A3: x14 + x24 + x34 + x44 <= 1
    c2B1: x22 + x23 + x24 + x25 <= 1
    c2B2: x32 + x33 + x34 + x35 <= 1
    c2B3: x42 + x43 + x44 + x45 <= 1
    c2AB1: x12 + x32 + x42 - x23 - x24 - x25 = 0
    c2AB2: x13 + x23 + x43 - x32 - x34 - x35 = 0
    c2AB3: x14 + x24 + x34 - x42 - x43 - x45 = 0

\ Constraint 3
    c3: 1846 x12 + 3616 x13 + 2811 x14 + 3524 x15 + 3434 x23 +
    2629 x24 + 2200 x25 + 3434 x32 + 1160 x34 + 5299 x35 + 2629 x42 +
    1160 x43 + 4495 x45 + 2200 x52 + 5299 x53 + 4495 x54 <= 10000
```

\ Constraint 4 handled through Bounds and Generals

\ Constraint 5

```
c5ui2uj2: 4 x22 <= 3
c5ui2uj3: u2 - u3 + 4 x23 <= 3
c5ui2uj4: u2 - u4 + 4 x24 <= 3
c5ui2uj5: u2 - u5 + 4 x25 <= 3
c5ui3uj2: u3 - u2 + 4 x32 <= 3
c5ui3uj3: 4 x33 <= 3
c5ui3uj4: u3 - u4 + 4 x34 <= 3
c5ui3uj5: u3 - u5 + 4 x35 <= 3
c5ui4uj2: u4 - u2 + 4 x42 <= 3
c5ui4uj3: u4 - u3 + 4 x43 <= 3
c5ui4uj4: 4 x44 <= 3
c5ui4uj5: u4 - u5 + 4 x45 <= 3
c5ui5uj2: u5 - u2 + 4 x52 <= 3
c5ui5uj3: u5 - u3 + 4 x53 <= 3
c5ui5uj4: u5 - u4 + 4 x54 <= 3
c5ui5uj5: 4 x55 <= 3
```

Bounds

```
2 <= u2 <= 5
2 <= u3 <= 5
2 <= u4 <= 5
2 <= u5 <= 5
x12 <= 1
x13 <= 1
x14 <= 1
x15 <= 1
x22 <= 1
x23 <= 1
x24 <= 1
x25 <= 1
x32 <= 1
x33 <= 1
x34 <= 1
x35 <= 1
x42 <= 1
x43 <= 1
x44 <= 1
x45 <= 1
x52 <= 1
x53 <= 1
x54 <= 1
x55 <= 1
```

Binary

```
x12 x13 x14 x15
x22 x23 x24 x25
```

x32 x33 x34 x35  
x42 x43 x44 x45  
x52 x53 x54 x55

Generals

u2 u3 u4 u5

End

## APPENDIX D – ITERATIVE PROCESS IDENTIFYING TOTAL COST CONSTRAINTS

The following tables catalogue the results of the iterative process to identify the desired total cost constraint values  $C_{max}$  for each 10 and 15-vertex test dataset used in the study. Once a model was built to optimally solve a dataset, the model was run repeatedly by reducing by one meter the total cost constraint from the previous solution path distance. The number of iterations run ranged from 12 to 33 for the 10-vertex sets, and from 22 to 56 for the 15-vertex sets. Generally a few iterations were completed beyond the observed lowest total cost constraint providing the desired number of vertices in the solution to ensure the lowest total cost constraint had been identified. The results of this iterative process for each dataset can be seen on the following pages.

**Table D1: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 10n\_set1**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
---	----------------------------	----------------------------------	--------------------------	----------------------------

10n set1 12000	404	11,690	Y	10
10n set1 10000	404	9,974	Y	10
10n set1 9973	397	9,911		9
10n set1 9910	397	9,579		9
10n set1 9578	397	9,578		9
10n set1 9577	393	8,579		9
10n set1 8579	393	8,490		9
10n set1 8489	393	8,212		9
10n set1 8211	386	8,094		8
10n set1 8093	386	7,817		8
10n set1 7816	386	7,816		8
10n set1 7815	350	7,516		8
10n set1 7515	343	7,120		7
10n set1 7119	271	6,822		6
10n set1 6821	187	6,543		5
10n set1 6542	169	4,961		6
10n set1 4960	162	4,492		5
10n set1 4491	138	4,088		4

*Note.* The yellow cell is the smallest  $C_{max}$  to include all 10 vertices in the solution. The green cell is the largest  $C_{max}$  to use 6 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 6 vertices in the solution.

**Table D2: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 10n\_set2**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of Vertices</b>
10n_set2 10000	372	9,608	Y	10
10n_set2 9607	365	9,557		9
10n_set2 9556	365	9,484		9
10n_set2 9483	352	9,260		9
10n_set2 9259	345	9,137		8
10n_set2 9136	306	9,119		8
10n_set2 9118	305	9,102		8
10n_set2 9101	299	8,372		7
10n_set2 8371	299	8,350		7
10n_set2 8349	299	8,349		7
10n_set2 8348	279	8,002		6
10n_set2 8001	239	7,967		6
10n_set2 7966	234	6,751		8
10n_set2 6750	234	5,754		8
<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of Vertices</b>
10n_set2 5753	234	5,202		8

10n_set2 5201	227	5,150	7
10n_set2 5149	227	5,078	7
10n_set2 5077	214	4,854	7
10n_set2 4853	207	4,731	6
10n_set2 4730	168	4,713	6
10n_set2 4712	167	4,696	6
10n_set2 4695	161	3,966	5
10n_set2 3965	161	3,943	5
10n_set2 3942	141	3,618	4

*Note.* The yellow cell is the smallest  $C_{max}$  to include all 10 vertices in the solution. The green cell is the largest  $C_{max}$  to use 6 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 6 vertices in the solution.

**Table D3: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 10n\_set3**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
10n set3 12000	326	11,589	Y	10
10n set3 11588	321	11,146		9
10n set3 11000	321	10,507		9
10n set3 10000	285	9,830		8
10n set3 9829	278	9,667		8
10n set3 9666	267	9,421		7
10n set3 9420	260	9,258		7
10n set3 9257	242	8,990		7
10n set3 8989	237	8,582		7
10n set3 8581	224	6,037		7
10n set3 6036	219	5,031		6
10n set3 5030	183	4,354		5
10n set3 4353	151	4,339		5
10n set3 4338	120	4,111		5
10n set3 4110	117	3,998		5
10n set3 3997	115	3,662		4

*Note.* The yellow cell is the smallest  $C_{max}$  to include all 10 vertices in the solution. The green cell is the only  $C_{max}$  to use 6 vertices in the solution.

**Table D4: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 10n\_set4**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
---	------------------------	------------------------------	----------------------	------------------------

10n set4 14000	501	13,936	Y	10
10n set4 13000	462	12,993		9
10n set4 12000	435	10,598		9
10n set4 11000	435	10,024		9
10n set4 10000	435	9,579		9
10n set4 9578	435	9,558		9
10n set4 9557	435	9,299		9
10n set4 9298	396	9,089		8
10n set4 9088	396	9,055		8
10n set4 9054	386	8,901		8
10n set4 8900	386	8,854		8
10n set4 8853	386	8,768		8
10n set4 8767	386	8,595		8
10n set4 8594	386	8,315		8
10n set4 8314	347	8,105		7
10n set4 8104	327	7,956		7
10n set4 7955	327	7,697		7
10n set4 7696	304	7,568		7
10n set4 7567	288	7,487		6
10n set4 7486	276	7,237		6
10n set4 7236	265	7,078		6
10n set4 7077	246	3,843		6
10n set4 3842	197	2,859		5

*Note.* The yellow cell is the smallest  $C_{max}$  to include all 10 vertices in the solution. The green cell is the largest  $C_{max}$  to use 6 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 6 vertices in the solution.

**Table D5: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 10n\_set5**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
10n set5 15000	327	14,139	Y	10
10n set5 14138	327	13,909	Y	10
10n set5 13908	327	13,819	Y	10
10n set5 13818	327	13,453	Y	10
10n set5 13452	327	12,858	Y	10
10n set5 12857	327	12,012	Y	10
10n set5 12011	327	11,936	Y	10
10n set5 11935	327	11,537	Y	10
10n set5 11536	320	10,161		9
<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
10n set5 11000	320	10,450		9



10n set5 10160	320	9,762	9
10n set5 10000	320	9,762	9
10n set5 9761	308	9,062	8
10n set5 9061	308	8,821	8
10n set5 8820	308	8,820	8
10n set5 8819	285	8,683	7
10n set5 8682	276	8,533	7
10n set5 8532	260	7,640	7
10n set5 7639	260	7,639	7
10n set5 7638	237	7,502	6
10n set5 7501	228	7,352	6
10n set5 7351	219	7,297	7
10n set5 7296	219	7,093	7
10n set5 7092	219	6,851	7
10n set5 6850	196	6,714	6
10n set5 6713	187	6,564	6
10n set5 6563	183	5,105	7
10n set5 5104	183	3,821	7
10n set5 3820	171	2,879	6
10n set5 2878	151	2,738	6
10n set5 2737	151	2,581	6
10n set5 2580	151	2,339	6
10n set5 2338	139	1,397	5

*Note.* The yellow cell is the smallest  $C_{max}$  to include all 10 vertices in the solution. The green cell is the largest  $C_{max}$  to use 6 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 6 vertices in the solution.

**Table D6: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 15n\_set1**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
15n set1 80000	530	27,050	Y	15
15n set1 30000	530	26,401	Y	15
15n set1 20000	530	18,076	Y	15
15n set1 18075	530	17,748	Y	15
15n set1 17747	530	17,650	Y	15
15n set1 17000	530	16,497		15
15n set1 16496	530	16,484		15
15n set1 16483	530	16,479		15
15n set1 16100	530	16,095		15
<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
15n set1 15000	529	14,703		14

15n set1 14000	510	13,997	11
15n set1 13500	504	13,461	11
15n set1 13000	495	12,901	12
15n set1 12500	476	12,359	10
15n set1 12358	472	12,215	11
15n set1 12214	472	12,101	11
15n set1 12100	472	12,039	11
15n set1 12038	470	11,894	10
15n set1 11893	470	11,823	10
15n set1 11821	470	11,761	10
15n set1 11760	460	11,755	11
15n set1 11754	458	11,477	10
15n set1 11476	453	11,378	9
15n set1 11377	442	11,245	10
15n set1 11244	442	11,034	10
15n set1 11033	440	10,967	9
15n set1 11000	440	10,967	9
15n set1 10966	440	10,756	9
15n set1 10755	435	10,687	9
15n set1 10686	435	10,615	9
15n set1 10614	423	10,373	8
15n set1 10000	407	9,888	9
15n set1 9887	405	9,610	8
15n set1 9609	390	9,505	8
15n set1 9504	388	9,227	7
15n set1 9226	380	9,215	8
15n set1 9214	380	9,163	8
15n set1 9162	380	9,153	8
15n set1 9152	368	9,122	7
15n set1 9121	368	8,911	7
15n set1 9000	368	8,911	7
15n set1 8910	363	8,780	7
15n set1 8779	363	8,770	7
15n set1 8769	352	8,426	8
15n set1 8425	350	8,359	7
15n set1 8358	350	8,148	7
15n set1 8147	335	8,043	7

**Note.** The green cell is the largest  $C_{max}$  to use 8 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 8 vertices in the solution.

**Table D7: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 15n\_set2**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
15n_set2 30000	699	29,616	Y	15
15n_set2 18000	699	17,953	Y	15
15n_set2 16000	699	15,280	Y	15
15n_set2 14000	684	13,919		13
15n_set2 13156	649	13,022		10
15n_set2 12000	605	11,565		12
15n_set2 11000	605	10,835		12
15n_set2 10000	590	9,886		10
15n_set2 9500	520	9,258		9
15n_set2 9257	501	9,255		9
15n_set2 9254	501	9,125		9
15n_set2 9124	501	8,962		9
15n_set2 8961	481	8,836		8
15n_set2 8835	481	8,825		8
15n_set2 7898	386	7,898		8
15n_set2 7897	379	7,630		7
15n_set2 7629	379	7,443		7
15n_set2 7442	379	7,442		7
15n_set2 7441	359	7,317		6
15n_set2 7316	359	7,305		6
15n_set2 7304	304	7,109		6
15n_set2 7108	304	7,108		6
15n_set2 7107	290	6,518		6
15n_set2 6517	270	6,382		5

*Note.* The green cell is the largest  $C_{max}$  to use 8 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 8 vertices in the solution.

**Table D8: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 15n\_set3**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
15n_set3 30000	668	27,851	Y	15
15n_set3 14000	641	13,968		14
15n_set3 12000	571	11,866		13
15n_set3 11000	521	10,737		12
15n_set3 10000	498	9,957		11
15n_set3 9956	491	9,884		10
15n_set3 9883	467	9,824		10
<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of Vertices</b>

15n_set3 9823	466	9,820	10
15n_set3 9819	460	9,751	9
15n_set3 9750	459	9,747	9
15n_set3 9746	454	9,686	11
15n_set3 9685	449	9,671	10
15n_set3 9670	447	9,613	10
15n_set3 9612	442	9,598	9
15n_set3 9597	431	9,569	10
15n_set3 9568	431	9,275	10
15n_set3 9274	431	9,028	10
15n_set3 9000	431	8,967	10
15n_set3 9027	431	8,967	10
15n_set3 8966	424	8,894	9
15n_set3 8893	400	8,834	9
15n_set3 8833	399	8,830	9
15n_set3 8829	393	8,761	8
15n_set3 8760	392	8,757	8
15n_set3 8756	375	8,734	8
15n_set3 8733	375	8,733	8
15n_set3 8732	368	8,697	8
15n_set3 8696	361	8,624	7
15n_set3 8623	346	8,503	9
15n_set3 8502	340	8,353	9
15n_set3 8352	340	8,292	9
15n_set3 8291	339	8,252	8
15n_set3 8251	339	8,241	8
15n_set3 8240	339	8,006	8
15n_set3 8005	308	7,743	7
15n_set3 7742	276	7,664	6
15n_set3 7663	276	7,607	6
15n_set3 7606	276	7,606	6
15n_set3 7000	224	6,984	8
15n_set3 6983	201	6,667	7
15n_set3 6666	197	6,290	7
15n_set3 6289	174	5,571	6

**Note.** The green cell is the largest  $C_{max}$  to use 8 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 8 vertices in the solution.

**Table D9: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 15n\_set4**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
15n_set4 30000	668	27,362	Y	15
15n_set4 20000	668	18,798	Y	15
15n_set4 18000	668	17,364	Y	15
15n_set4 16000	640	15,875		12
15n_set4 14000	546	13,955		11
15n_set4 12000	397	11,921		11
15n_set4 11000	397	10,800		11
15n_set4 10799	397	10,763		11
15n_set4 10762	385	10,463		10
15n_set4 10462	385	10,271		10
15n_set4 10270	385	9,879		10
15n_set4 9878	370	9,617		9
15n_set4 9616	353	9,237		9
15n_set4 9236	341	9,122		8
15n_set4 9121	341	9,083		8
15n_set4 9082	341	8,805		8
15n_set4 8804	341	8,353		8
15n_set4 8352	326	8,091		7
15n_set4 8090	287	8,019		7
15n_set4 8018	286	7,937		6
15n_set4 7936	272	7,757		6
15n_set4 7756	271	7,675		5

*Note.* The green cell is the largest  $C_{max}$  to use 8 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 8 vertices in the solution.

**Table D10: Results of Iterative Process to Identify Target Total Cost Constraint Values for Dataset 15n\_set5**

<b>Dataset &amp; <math>C_{max}</math></b>	<b>Objective Value</b>	<b>Solution Distance (m)</b>	<b>Max vertices?</b>	<b>No. of vertices</b>
15n_set5 20000	634	19,026	Y	15
15n_set5 16000	623	15,728		13
15n_set5 14000	594	13,949		12
15n_set5 12000	559	11,869		11
15n_set5 11868	536	11,528		11
15n_set5 11527	531	11,375		11
15n_set5 11374	527	11,289		10
15n_set5 11288	527	11,242		10
15n_set5 11241	512	11,094		10
15n_set5 11093	505	10,482		11

Dataset & $C_{max}$	Objective Value	Solution Distance (m)	Max vertices?	No. of vertices
15n_set5 10481	505	10,228		11
15n_set5 10227	496	10,001		10
15n_set5 10000	496	9,648		10
15n_set5 9647	473	9,601		10
15n_set5 9600	473	9,369		10
15n_set5 9368	464	9,081		9
15n_set5 9080	464	9,068		9
15n_set5 9067	464	9,021		9
15n_set5 9020	449	8,873		9
15n_set5 8872	440	8,857		9
15n_set5 8856	431	8,754		8
15n_set5 8753	431	8,510		8
15n_set5 8509	417	8,306		8
15n_set5 8305	403	7,984		9
15n_set5 7983	385	7,658		8
15n_set5 7657	380	7,399		9
15n_set5 7398	371	7,051		8
15n_set5 7050	356	6,856		8
15n_set5 6855	353	6,748		7
15n_set5 6747	338	6,665		7
15n_set5 6664	338	6,595		7
15n_set5 6594	338	6,493		7
15n_set5 6492	324	6,289		7
15n_set5 6288	301	5,497		8
15n_set5 5496	301	5,261		8
15n_set5 5260	301	5,247		8
15n_set5 5246	301	5,029		8
15n_set5 5028	292	4,899		7
15n_set5 4898	292	4,856		7

*Note.* The green cell is the largest  $C_{max}$  to use 8 vertices in the solution. The blue cell is the smallest  $C_{max}$  to use 8 vertices in the solution.

## REFERENCES

- Cao, B., Sun, M., & Macleod, C. (1999). Applying GIS and Combinatorial Optimization to Fiber Deployment Plans. *Journal of Heuristics*, 5(4), 385–402.  
<https://doi.org/10.1023/A:1009628321600>
- Chao, I.-M., Golden, B. L., & Wasil, E. A. (1996). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3), 475–489. [https://doi.org/10.1016/0377-2217\(95\)00035-6](https://doi.org/10.1016/0377-2217(95)00035-6)
- Church, R. L. (2002). Geographical information systems and location science. *Computers & Operations Research*, 29(6), 541–562. [https://doi.org/10.1016/S0305-0548\(99\)00104-5](https://doi.org/10.1016/S0305-0548(99)00104-5)
- Church, R. L., & Sorensen, P. (1994). Integrating Normative Location Models into GIS: Problems and Prospects with the p-median Model (94-5). *EScholarship*. Retrieved from <http://escholarship.org/uc/item/7nz7762k>
- Curtin, K. M., Voicu, G., Rice, M. T., & Stefanidis, A. (2014). A Comparative Analysis of Traveling Salesman Solutions from Geographic Information Systems. *Transactions in GIS*, 18(2), 286–301. <https://doi.org/10.1111/tgis.12045>
- Duque, D., Lozano, L., & Medaglia, A. L. (2015). An exact method for the biobjective shortest path problem for large-scale road networks. *European Journal of Operational Research*, 242(3), 788–797.  
<https://doi.org/10.1016/j.ejor.2014.11.003>

- Eccles, D. W., Walsh, S. E., & Ingledew, D. K. (2002). The use of heuristics during route planning by expert and novice orienteers. *Journal of Sports Sciences*, 20(4), 327–337. <https://doi.org/10.1080/026404102753576107>
- Erdoğan, G., Cordeau, J.-F., & Laporte, G. (2010). The Attractive Traveling Salesman Problem. *European Journal of Operational Research*, 203(1), 59–69. <https://doi.org/10.1016/j.ejor.2009.06.029>
- Feillet, D., Dejax, P., & Gendreau, M. (2005). Traveling Salesman Problems with Profits. *Transportation Science*, 39(2), 188–205. <https://doi.org/10.1287/trsc.1030.0079>
- Gendreau, M., Laporte, G., & Semet, F. (1998). A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, 106(2), 539–545. [https://doi.org/10.1016/S0377-2217\(97\)00289-0](https://doi.org/10.1016/S0377-2217(97)00289-0)
- Golden, Levy, L., & Vohra, R. (1987). The Orienteering Problem. *Naval Research Logistics*, 34, 307–318.
- Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering Problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2), 315–332. <https://doi.org/10.1016/j.ejor.2016.04.059>
- Gurobi Optimization. (n.d.). Mathematical Programming Solver | Gurobi. Retrieved September 28, 2018, from <http://www.gurobi.com/products/gurobi-optimizer>
- International Rogaining Federation. (2017, January 1). Rules of Rogaining.



- Kataoka, S., & Morito, S. (1988). An Algorithm for Single Constraint Maximum Collection Problem. *Journal of the Operations Research Society of Japan*, 31(4), 515–531. <https://doi.org/10.15807/jorsj.31.515>
- Laporte, G., & Martello, S. (1990). The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2), 193–207. [https://doi.org/10.1016/0166-218X\(90\)90100-Q](https://doi.org/10.1016/0166-218X(90)90100-Q)
- Mei, X. (2015). *Approximating the Length of Vehicle Routing Problem Solutions Using Complementary Spatial Information* (Dissertation). George Mason University. Retrieved from <http://mars.gmu.edu/handle/1920/9623>
- Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer Programming Formulation of Traveling Salesman Problems. *J. ACM*, 7(4), 326–329. <https://doi.org/10.1145/321043.321046>
- Okabe, A., & Okunuki, K. (2013). SANET: Spatial Analysis along Networks (Version 4.1). Tokyo, Japan. Retrieved from [http://sanet.csis.u-tokyo.ac.jp/download/manual\\_standalone.pdf](http://sanet.csis.u-tokyo.ac.jp/download/manual_standalone.pdf)
- Orienteering USA. (2018). Orienteering Formats. Retrieved November 6, 2018, from <https://www.orienteeringusa.org/new-o/what-orienteering/o-formats>
- Palomo-Martínez, P. J., Salazar-Aguilar, M. A., & Albornoz, V. M. (2017). Formulations for the orienteering problem with additional constraints. *Annals of Operations Research*, 258(2), 503–545. <https://doi.org/10.1007/s10479-017-2408-4>

- Ramesh, R., & Brown, K. M. (1991). An efficient four-phase heuristic for the generalized orienteering problem. *Computers & Operations Research*, 18(2), 151–165.  
[https://doi.org/10.1016/0305-0548\(91\)90086-7](https://doi.org/10.1016/0305-0548(91)90086-7)
- Ramesh, R., Yoon, Y.-S., & Karwan, M. H. (1992). An Optimal Algorithm for the Orienteering Tour Problem. *ORSA Journal on Computing*, 4(2), 155.
- Righini, G., & Salani, M. (2009). Decremental state space relaxation strategies and initialization heuristics for solving the Orienteering Problem with Time Windows with dynamic programming. *Computers & Operations Research*, 36(4), 1191–1203. <https://doi.org/10.1016/j.cor.2008.01.003>
- Tsiligirides, T. (1984). Heuristic Methods Applied to Orienteering. *The Journal of the Operational Research Society*, 35(9), 797–809. <https://doi.org/10.2307/2582629>
- U.S. Census Bureau QuickFacts: Fairfax city, Virginia (County). (n.d.). Retrieved February 20, 2018, from  
<https://www.census.gov/quickfacts/fact/table/fairfaxcityvirginiacounty/PST045216>
- Vansteenwegen, P., Souffriau, W., & Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, 209(1), 1–10.  
<https://doi.org/10.1016/j.ejor.2010.03.045>
- Verbeeck, C., Sörensen, K., Aghezzaf, E.-H., & Vansteenwegen, P. (2014). A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, 236(2), 419–432.  
<https://doi.org/10.1016/j.ejor.2013.11.038>

Victorian Rogaining Association. (2016). What is Rogaining? Retrieved November 6, 2018, from <https://vra.rogaine.asn.au/getting-started/what-is-rogaining>

## **BIOGRAPHY**

John K Robinson graduated from Thomas Jefferson High School for Science and Technology, Alexandria, Virginia, in 1997. He received his Bachelor of Arts in Government from the University of Virginia in 2001 and his Master of Arts in Education and Human Development from George Washington University in 2010. He has worked as a Legislative Aide at the U.S. Senate, an Assistant Director of Undergraduate Admissions at George Washington University, and Academy Coordinator for the National Automobile Dealers Association. He enjoys being in the outdoors, listening to music, and crime/mystery television shows. He plans to receive his Master of Science in Geographic and Cartographic Sciences from George Mason University in 2018.