2016

Routing Problems for Unmanned Surface Vehicles with Limited Battery Life

Joshua Taylor Margolis
Lehigh University

Routing Problems for Unmanned Surface Vehicles With Limited Battery Life

by

Joshua T. Margolis

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science in

Industrial and Systems Engineering

Lehigh University

May 2016

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

Date

Lawrence V. Snyder, Thesis Advisor

Tamás Terlaky, Chairperson of Department

Acknowledgements

I would first like to thank my thesis advisor Dr. Lawrence V. Snyder. He allowed this paper to be my own work, but would always provide aid and insight when needed. I would like to extend my sincere gratitude for all that he has taught me.

Secondly, I would like to thank the faculty of the Industrial and Systems Engineering Department at Lehigh University for the numerous learning experiences they provided. In particular, I would like to thank Dr. Frank E. Curtis for his help with LaTeX, but also for providing his counsel during this experience. Moreover, I would like to thank Dr. Martin Takáč for his help with using the department cluster.

Finally, I would like to express my profound gratitude to my friends, to my family, and to my girlfriend Kathleen for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Tables **vii**

List of Figures **viii**

Abstract **1**

1 Introduction **2**

1.1 The Vehicle Routing Problem 2

1.2 Heuristics 4

1.3 The Covering Tour Problem 5

1.4 Velocity Routing 6

2 The Multi-Vehicle Covering Tour Problem **8**

2.1 Notation 8

2.2 Formulation I 11

2.3 Formulation II 13

3 Heuristics **15**

3.1 Modified Sweep Heuristic 15

3.1.1 Applying the Sweep Algorithm to Formulation I (Sec. 2.2) 15

3.1.2 Applying the Sweep Algorithm to Formulation II (Sec. 2.3) 16

3.2 Modified Savings Heuristic 20

3.2.1 Applying the Savings Algorithm to Formulation I (Sec. 2.2) 21

3.2.2 Applying the Savings Algorithm to Formulation II (Sec. 2.3) 22

4 Velocity Routing **23**

4.1 Determining Vehicle Velocity for a Known Route 23

4.2 The Multi-Vehicle Covering Tour Problem with Velocities 25

5 Computational Results **29**

5.1 Routing 29

5.1.1 Formulation I 30

5.1.2 Formulation II 35

5.2 Routing with Velocity 41

6 Conclusion **44**

Bibliography **46**

Biography **49**

List of Tables

2.1 Notation 9

5.1 Description of problem instances 29

5.2 Results for Formulation I 30

5.3 Results for Formulation II 36

5.4 Parameter modifications for velocity routing 42

5.5 Route and velocity results for Instance 4_8 42

5.6 Route and velocity results for Instance 8_16 42

List of Figures

2.1 Schematic of Eq. (2.1) 10

2.2 Schematic of Eq. (2.2) 11

3.1 Schematic of sweep heuristic for Formulation I 19

5.1 Site layout and optimal/heuristic solutions for Instance 4_0_8 for Formulation I. . . 31

5.2 Site layout and optimal/heuristic solutions for Instance 4_4_16 for Formulation I. . 32

5.3 Site layout and optimal/heuristic solutions for Instance 6_6_12 for Formulation I. . 33

5.4 Site layout and optimal/heuristic solutions for Instance 12_14_21 for Formulation I. 34

5.5 Comparison of CPLEX, Modified Savings, and Modified Sweep heuristics for Formulation I. 35

5.6 Site layout and optimal/heuristic solutions for Instance 4_0_8 for Formulation II. . 37

5.7 Site layout and optimal/heuristic solutions for Instance 4_4_16 for Formulation II. . 38

5.8 Site layout and optimal/heuristic solutions for Instance 6_6_12 for Formulation II. . 39

5.9 Site layout and optimal/heuristic solutions for Instance 12_14_21 for Formulation II. 40

5.10 Comparison of CPLEX, Modified Savings and Modified Sweep heuristics for Formulation II. 41

5.11 Map of vehicle routes for instance 4_8 43

Abstract

Given a set of locations (i.e. bridges, bays, docks, etc.) that must be inspected and a set of waypoints, we design and implement a model to route a fleet of unmanned surface vehicles via a set of waypoints that allow the aforementioned locations to be surveilled. Furthermore, the velocity at which the vehicles traverse each part of the route is dependent upon the level of surveillance required for each site. More specifically, the model constructs the optimal set of routes for at most K unmanned surface vehicles that minimizes the fleet's total distance, subject to distance, battery life, and site number constraints, while ensuring that a set of sites are covered during the tours. In addition, the model also determines the velocity of each vehicle along each arc of the tour, where the velocity is dependent upon the importance of the sites that are covered along that arc. Lastly, we modify, design, and implement heuristics to construct feasible solutions.

Chapter 1

Introduction

The cost of transportation is a major strategic driver throughout the supply chain. This category of expenditures encompasses the price of fuel, the fixed cost for deploying a vehicle, and the cost of labor. In particular, suppliers want to curtail the total shipment cost without diminishing revenue. One prudent method is to optimize the routes for the fleet of vehicles tasked with delivering the products, such that the collective distance traveled, and thus the transportation cost, is minimized. Though there may be restrictions placed upon the vehicles regarding route length or capacity, this overall concept is classified as a *vehicle routing problem*.

1.1 The Vehicle Routing Problem

Routing problems can be modeled as a network, where the nodes represent the locations of both producers and consumers, while the edges represent the possible transportation roads. Thus, the *vehicle routing problem* (VRP) is an integer programming problem that seeks to find the optimal set of routes for a set of vehicles to deliver services to a set of customers; each route must begin and end at a given node, called the depot [26].

In general, there are two classes of *vehicle routing problems*. Node routing problems, such as package delivery, bus routing, and utility and service maintenance, involve routing vehicles to either deliver or pick up goods/services from distinct locations. Arc routing problems, like snow removal, postal delivery and waste collection, on the other hand, involve routing vehicles to serve or cover links in a network or streets in a city. The former category is concerned with the nodes

of the network, while the latter deals with the edges of the graph [7].

Node routing problems can be comprised of an array of characteristics. In [26], the authors outline a few variations:

- *Time windows* during which each vehicle must visit a specific node or arrive back at the depot (utility/service maintenance) .
- *Backhauls*, where vehicles deliver products to some customers and pick up product from others.
- *Pickups and deliveries*, in which certain customers require both pickups and deliveries (public transportation).
- *Periodic* models in which customers are visited a fixed number of times per interval.

Moreover, arc routing problems involve traversing along an edge of the network; the number of traversals, though, is dependent upon the application. For example, snow removal vehicles might want to clear specific roads multiple times per day [21], but sanitation vehicles need only travel a street once a week. If each arc must be traversed only once, it is called a *Chinese Postman Problem* [7].

Furthermore, the direction of travel may be important. The postal service oftentimes traverses a single road twice per day, once in either direction, in order to deliver mail to both sides of the street without disrupting traffic. Likewise, snow removal vehicles will traverse major roadways multiple times, whereas local roads are subject to only a single pass in a single direction [7]. Both of these examples introduce limitations on the routes by adding the direction of travel as a constraint. In general, there are three subcategories for arc routing problems: a *directed* network is when the direction a link must be traversed is constrained; an *undirected* network is when the direction is not mandated; and a *mixed* network is a combination of the previous two. As it turns out, the latter is the more difficult to solve [7].

The VRP closely resembles the *traveling salesman problem* (TSP), since both are combinatorial optimization problems with similar objectives and problem structures; however, the TSP focuses on routing an individual entity, whereas the VRP pertains to multiple vehicles. Furthermore, the

VRP not only decides the optimal customer order along the route, but it must also determine which customers are assigned to specific tours; hence, it is difficult to solve to optimality. As a result, it is a more challenging optimization problem than the TSP [26].

The *vehicle routing problem* was first introduced by Danztig and Ramser [6], and aimed to determine the optimal routes for a fleet of gasoline delivery trucks tasked with transporting fuel between a central depot and multiple service stations. Labelled as the *Truck Dispatching Problem*, the "best solution" was obtained by utilizing methods of linear programming after relaxing the integrality constraints [6]. Since then, multiple other algorithms have been designed to solve the VRP.

1.2 Heuristics

In fact, there are a plethora of heuristics that have been designed for specific variants of the VRP, whereas some are more general. Laporte [20] provides an overview of both exact and approximate algorithms. In this section, we focus on the generic algorithms. According to [26], the *Clark--Wright savings heuristic* [2] is not only one such algorithm, it is one of the best. It begins by placing each node on its own route and then merges routes in a systematic way. That is, the alteration is made if and only if combining two routes reduces the total cost and maintains feasibility. The concept of feasibility, moreover, involves ensuring that any distance, customer, capacity or fuel limitation is not violated.

Another heuristic, known as the *sweep heuristic* [14, 27, 28], generates groups of nodes by rotating a half-line extending from the depot in a clockwise or counterclockwise motion. When the ray intersects a node, it is added to the current route unless doing so creates an infeasibility. In that regard, the current route is terminated by connecting the tour back to the depot and beginning a new route. Once the nodes are distributed into clusters, routes can be improved by solving the TSP on each subset; hence, it is an example of a *two-phase method* [26].

Both the savings algorithm and sweep algorithm are deterministic in that repeating either algorithm will produce the same result. Furthermore, neither algorithm possesses the capability to look a few steps ahead; as a result, each may produce an optimality gap [7]. Randomizing these heuristics can produce improved solutions, though doing so will also increase the total run-time

[7]. Moreover, these heuristics can be adapted to find feasible solutions for a multitude of VRP variations.

1.3 The Covering Tour Problem

One such extension of the VRP is the *covering tour problem* (CTP). The CTP aims to construct the minimum length tour for a fleet of vehicles, where V is the set of vertices that can be visited, $T \subseteq V$ is the set of vertices that must be visited, and W is the set of vertices that must be covered, i.e., every vertex of W must lie within a distance r from a vertex on the tour [13]. According to Gendreau et al. [13], the CTP was first introduced by Current [3] and was formulated in Current and Schilling [4]. Current and Schilling [5] model an extension of the CTP as a bicriterion optimization problem called the *maximal covering tour problem* (MCTP). The objective is to maximize the total demand covered along a given route. Another application of the CTP involves locating a set of post boxes to not only minimize the cost of the corresponding routes, but also to ensure that they are easily accessible for every customer [19]. Another application is to study how to locate medical facilities in such a way that they are easily accessible by patients and that supplies can be easily delivered [8, 17, 23]. Revelle and Laporte [24] label the CTP as the *Traveling Circus Problem*, where they sought to route a traveling circus through a set of potential locations such that the stops were a reasonable distance for patrons in the surrounding areas. Gendreau et al. [13] formulate the CTP as a linear, integer programming problem and devise an exact branch and cut algorithm to solve it. Jozefowiez et al. [18] generalize the CTP by reformulating it as a bi-objective problem; they replace the covering constraints with an additional objective. Moreover, Sahraeian and Ebrahimi [25] examine the *allocated maximal backup covering tour problem* (AMBCTP), which is a variation of the standard CTP with the added objective of maximizing the number of sites that are covered more than once.

Furthermore, the CTP can be expanded to account for a fleet of vehicles, as in police patrol or coastal surveillance. This extension is called the *multi-vehicle covering tour problem* (m-CTP), where at most m vehicles are used. Surveillance vehicles, like police patrol vehicles, must depart and return to a depot, while traversing roads to visit high risk locales and have lower risk locations within view. That is, police must visit those areas that pose a greater security risk, but also want

to surveil areas that may not require as thorough an inspection. The police department thus must route a fleet of vehicles for this purpose with the objective of minimizing the total distance in such a way that there is a level of uniformity to each route.

Hachicha et al. [16] formulate the m-CTP as an integer programming problem and develop a modified sweep heuristic, modified savings heuristic, and a modified route-first, cluster-second heuristic. Furthermore, Oliveira et al. [22] modify the aforementioned model and heuristics to construct routes for urban patrolling; they fix the number of routes in order to ensure that every vehicle is used.

We utilize these papers to formulate a variant of the m-CTP. That is, we modify the model proposed in [16] to deal with unmanned surface vehicles (USVs). These vehicles can be operated remotely or are run autonomously. As such, they are battery powered and have a limited energy supply.

1.4 Velocity Routing

With respect to patrol routing, it is important to not only thoroughly surveil the designated areas, but to also return to the depot before depleting the battery. Certainly some targets require a more rigorous inspection, while others only need a passing glance. Though the speed at which these vehicles pass locales is an important decision, it is limited by the turning angles along the route; the greater the turn, the more energy is needed and the lower the attainable speed becomes. Note that routing with an energy budget and routing to minimize the total time/maximize the total speed are not the same thing, though they can be related. In [12], the authors consider how fast a vehicle can travel through a turn by studying the fastest path for a nonholonomic agent (a vehicle whose state is dependent upon the path taken to achieve it) given a minimal turn radius. In [11], the authors expand upon the *Dubins car problem* by introducing direction dependent speeds. Furthermore, [9] seeks to find the quickest path in a direction dependent medium, such as water, by analyzing the *fastest-path finding problem*, by utilizing the maximum speed a vessel can travel while turning with respect to its current heading, from [10].

With developing technologies, though, the use of unmanned or remote control vehicles for both coastal and midland surveillance is now a possibility. In particular, coastal surveillance is a means for patrolling the coastline to detect, deter, and respond to a range of threats; it can serve a range of objectives. Whether it be bomb detection, crime prevention, locating enemy combatants, or emergency response, coastal surveillance is a practical and beneficial system. Specifically, using a fleet of USVs, we are tasked with constructing efficient routes to surveil specified locations. Each USV, though, operates on battery power, and thus must return to the central depot prior to draining its energy reserves.

We modify the multi-vehicle covering tour problem outlined in [16] to formulate an integer programming model to route a fleet of unmanned surface vehicles tasked with coastal surveillance. In addition to limiting the route length, number of sites per tour, and the total number of routes, the vehicles have a finite supply of battery life. Furthermore, we seek to determine the speed a vehicle traverses every arc based upon the level of security required for the sites they cover. Unlike [16] and [22] where populations or locales are only covered if a route makes a stop within a predetermined distance, we expand the definition of covering to allow for populations or locales to be within a specific distance of any point of the route. That is, we apply the definition of covering to the arcs of the route, not just the nodes.

The remainder of this thesis is organized as follows. In Chapter 2 we discuss the two variations of the model. The heuristics are presented in Chapter 3. Chapter 4 we describe velocity routing for a known tour and extend the models from Chapter 2 to include velocity as a decision variable. We present the computational results and overall summary in Chapter 5 and Chapter 6, respectively.

Chapter 2

The Multi-Vehicle Covering Tour Problem

In this chapter we introduce a multi-vehicle covering tour model. Specifically, the model constructs the optimal routes for a fleet of unmanned surface vehicles that minimizes the fleet's total distance traveled, while ensuring that a set of sites are covered during the tours. The individual vehicle routes, moreover, are subject to both limitations based upon the physical capabilities of the vehicles and limitations designed to homogenize the individual routes. In particular, each vehicle's route is constrained by a maximal route length, finite battery life, and maximal number of stops.

2.1 Notation

Let $K = \{1, \ldots, k\}$ be the set vehicles and $V = \{i \mid i \in [0, \ldots, N]\}$ the set of potential locations at which vehicles may stop. Furthermore, let W represent the set of sites that must be covered. All vehicles are identical, and should they be routed, must depart from and return to the depot, denoted as 0; we do not require every vehicle to be utilized. For notational purposes, we define $V^* = V \backslash \{0\}$. Moreover, let p be the maximum number of sites each vehicle is permitted to visit (excluding the depot), q the maximum distance a vehicle can travel, r the coverage radius, and E the maximum energy available for each vehicle. Table 2.1 summarizes the notation.

Given $\{a, b\} \in V \cup W$, we define $c_{a,b}$ as the distance between site a and site b. Moreover, we

Sets	**Description**
0	the depot
V	set of locations at which vehicles *may* stop
W	set of locations that *must* be covered
T	set of locations at which vehicles *must* stop where $T \subseteq V$
V^*	$V \backslash \{0\}$
T^*	$T \backslash \{0\}$
K	set of m USVs

Parameters	
p	maximum number of sites that can each USV can visit (excluding the depot)
q	maximum distance a USV can travel
r	coverage radius
$c_{i,j}$	distance between point i and point j
$h_{i,j,w}$	$= 1$ if w in W is within a distance r from the arc (i, j); 0 otherwise
$e_{i,j,m}$	energy required to go from point i to point j if point m is the subsequent stop (where $j \neq v_0$)
E	maximum energy available for each USV

Decision Variables ($\{i, j, m\} \in V$)	
$x_{i,j,k}$	$= 1$ if vehicle k uses arc $i \rightarrow j$; 0 otherwise
$y_{i,k}$	$= 1$ if vehicle k stops at site i; 0 otherwise
$z_{i,j,m,k}$	$= 1$ if vehicle k uses the route $i \rightarrow j \rightarrow m$; 0 otherwise

Table 2.1: Notation

need to account for the fact that a site $w \in W$ can be covered by vehicle k during its route. That is, if w is within a distance r to any point of the route, then it is covered by our definition. Hence, given $\{i, j\} \in V$ and $w \in W$, we define

$$h_{i,j,w} = \begin{cases} 1, & \text{if } w \in W \text{ is within a distance } r \text{ from the arc } (i,j) \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Specifically, Eq. (2.1) is set to 1 if the projection of w onto the line segment with endpoints i and j is within a distance r. Figure 2.1 provides a visual representation of this definition.

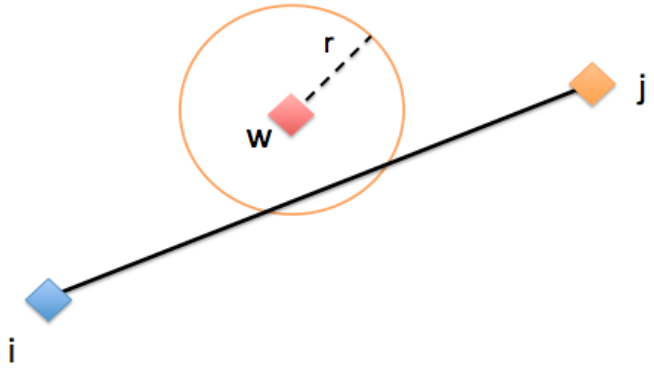


Figure 2.1: Schematic of Eq. (2.1)

Note that these values are precomputed. Lastly, we express the energy required to travel from site i to site j, given that site m is the subsequent stop, as $e_{i,j,m}$.

The distance between and two nodes is assumed to be known. Further, we assume there is negligible wave activity on the water's surface and that the energy used for a given sequence of stops is determined by the distance traveled and the angles of the turns along the route. That is,

$$e_{i,j,m} = l_1(c_{i,j} + c_{j,m}) + l_2(\pi - \theta), \tag{2.2}$$

where

$$\theta = \frac{s_{i,j} \cdot s_{j.m}}{c_{i,j} \cdot c_{j,m}}, \tag{2.3}$$

$\{l_1, l_2\} \in \mathbb{R}$, and $s_{i,j}$ is the vector from node i to node j. Figure 2.2 illustrates this definition.

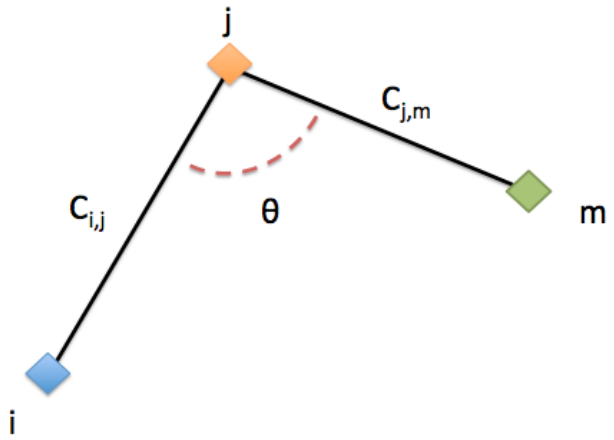


Figure 2.2: Schematic of Eq. (2.2)

There are three sets of decision variables:

$$y_{i,k} = \begin{cases} 1, & \text{if vehicle } k \text{ stops at site } i \\ 0, & \text{otherwise,} \end{cases} \tag{2.4}$$

$$x_{i,j,k} = \begin{cases} 1, & \text{if vehicle } k \text{ travels from site } i \text{ to site } j \\ 0, & \text{otherwise,} \end{cases} \tag{2.5}$$

$$z_{i,j,m,k} = \begin{cases} 1, & \text{if vehicle } k \text{ travels from site } i \text{ to site } j \text{ and then to site } m \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

2.2 Formulation I

Consider the requirement that a set of sites must be visited. That is, we must optimally route a set of k vehicles to cover all sites $w \in W$ wherein the vehicles must visit specific pre-determined sites and have the potential to visit remaining sites as needed. Specifically, let T be the set of sites the fleet of vehicles must collectively visit, where $T \subseteq V$ and $\{0\} \in T$. Furthermore, we denote $T^* = T \backslash \{0\}$. In summary, we must route the set of vehicles such that every site $t \in T^*$ is visited exactly once and every site $v \in V \backslash T$ is visited no more than once, such that every site $w \in W$ is covered.

Hence, the formulation is:

$$\text{minimize} \quad \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{i,j}x_{i,j,k} \tag{2.7}$$

$$\sum_{k\in K}\sum_{i\in V}\sum_{j\in V\setminus\{i\}} h_{i,j,w}\ x_{i,j,k} \geq 1 \qquad \forall\ w \in W \tag{2.8}$$

$$\sum_{k\in K} y_{i,k} \leq 1 \qquad \forall\ i \in V\setminus T \tag{2.9}$$

$$\sum_{i\in V}\sum_{j\in V\setminus\{i\}} c_{i,j}x_{i,j,k} \leq q \qquad \forall\ k \in K \tag{2.10}$$

$$\sum_{i\in V^*} y_{i,k} \leq p \qquad \forall\ k \in K \tag{2.11}$$

$$\sum_{k\in K} y_{i,k} = 1 \qquad \forall\ i \in T^* \tag{2.12}$$

$$y_{0,k} \leq 1 \qquad \forall\ k \in K \tag{2.13}$$

$$\sum_{j\in V\setminus\{i\}} x_{i,j,k} = y_{i,k} \qquad \forall\ i \in V, k \in K \tag{2.14}$$

$$\sum_{j\in V\setminus\{i\}} x_{j,i,k} = y_{i,k} \qquad \forall\ i \in V, k \in K \tag{2.15}$$

$$\sum_{i\in V}\sum_{j\in V^*\setminus\{i\}}\sum_{m\in V\setminus\{j\}} e_{i,j,m}z_{i,j,m,k} \leq E \qquad \forall\ k \in K \tag{2.16}$$

$$z_{i,j,m,k} \leq x_{i,j,k} \qquad \forall\ i \in V, j \in V^*\setminus\{i\}, m \in V\setminus\{j\}, k \in K \tag{2.17}$$

$$z_{i,j,m,k} \leq x_{j,m,k} \qquad \forall\ i \in V, j \in V^*\setminus\{i\}, m \in V\setminus\{j\}, k \in K \tag{2.18}$$

$$z_{i,j,m,k} \geq x_{i,j,k} + x_{j,m,k} - 1 \qquad \forall\ i \in V, j \in V^*\setminus\{i\}, m \in V\setminus\{j\}, k \in K \tag{2.19}$$

$$\sum_{i\in S}\sum_{j\in V\setminus S} x_{i,j,k} + \sum_{i\in S}\sum_{j\in V\setminus S} x_{j,i,k} \geq 2y_{h,k} \qquad \forall\ k \in K, S \subseteq V^*, h \in S \tag{2.20}$$

$$x_{i,j,k} \in \{0,1\} \qquad \forall\ i \in V, j \in V, k \in K \tag{2.21}$$

$$y_{i,k} \in \{0,1\} \qquad \forall\ i \in V, k \in K \tag{2.22}$$

$$z_{i,j,m,k} \in \{0,1\} \qquad \forall\ i \in V, j \in V, m \in V, k \in K \tag{2.23}$$

$$\tag{2.24}$$

The objective (2.7) is to minimize the total distance traveled for the entire fleet. Constraints

(2.8) ensure that each site in W is covered at least once. Constraints (2.9) dictate that each site in $V \backslash T$ is visited at most once. Constraints (2.10) limit the length of each route. Constraints (2.11) specify that each vehicle can visit at most p sites (excluding the base). Constraints (2.12) ensure that each site that must be visited is only visited by one vehicle. Constraints (2.30) make sure that each vehicle goes to the base if it is deployed. Constraints (2.14) ensure that there is only one route leaving i for vehicle k if vehicle k visits point i. Constraints (2.15) ensure that there is only one route entering i for vehicle k if vehicle k visits point i. Constraints (2.16) are the energy constraints, making sure that vehicle k's route does not exceed the battery capacity. Constraints (2.17)-(2.19) ensure that $z_{i,j,m,k} = 1$ if and only if vehicle k goes from $i \rightarrow j$ and then $j \rightarrow m$. Constraints (2.20) are the subtour elimination constraints. The remaining constraints correspond to the standard integrality conditions for the decision variables.

2.3 Formulation II

Instead of requiring a set of sites to be visited, we instead allow the model to route the vehicles to any of the potential locations in order to ensure that every site $w \in W$ is covered. That is, we set $T = \emptyset$ and continue to require that each vehicle both leaves from and returns to the depot. As a result, we have the following model.

$$\text{minimize} \quad \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j,k} \tag{2.25}$$

$$\sum_{k \in K} \sum_{i \in V} \sum_{j \in V \backslash \{i\}} h_{i,j,w} \; x_{i,j,k} \geq 1 \qquad \forall \; w \in W \tag{2.26}$$

$$\sum_{k \in K} y_{i,k} \leq 1 \qquad \forall \; i \in V^* \tag{2.27}$$

$$\sum_{i \in V} \sum_{j \in V \backslash \{i\}} c_{i,j} x_{i,j,k} \leq q \qquad \forall \; k \in K \tag{2.28}$$

$$\sum_{i \in V^*} y_{i,k} \leq p \qquad \forall \; k \in K \tag{2.29}$$

$$y_{0,k} \leq 1 \qquad \forall \; k \in K \tag{2.30}$$

$$\sum_{j \in V \backslash \{i\}} x_{i,j,k} = y_{i,k} \qquad \forall \; i \in V, k \in K \tag{2.31}$$

$$\sum_{j \in V \backslash \{i\}} x_{j,i,k} = y_{i,k} \qquad \forall\, i \in V, k \in K \tag{2.32}$$

$$\sum_{i \in V} \sum_{j \in V^* \backslash \{i\}} \sum_{m \in V \backslash \{j\}} e_{i,j,m} z_{i,j,m,k} \leq E \qquad \forall\, k \in K \tag{2.33}$$

$$z_{i,j,m,k} \leq x_{i,j,k} \qquad \forall\, i \in V, j \in V^* \backslash \{i\}, m \in V \backslash \{j\}, k \in K \tag{2.34}$$

$$z_{i,j,m,k} \leq x_{j,m,k} \qquad \forall\, i \in V, j \in V^* \backslash \{i\}, m \in V \backslash \{j\}, k \in K \tag{2.35}$$

$$z_{i,j,m,k} \geq x_{i,j,k} + x_{j,m,k} - 1 \qquad \forall\, i \in V, j \in V^* \backslash \{i\}, m \in V \backslash \{j\}, k \in K \tag{2.36}$$

$$\sum_{i \in S} \sum_{j \in V \backslash S} x_{i,j,k} + \sum_{i \in S} \sum_{j \in V \backslash S} x_{j,i,k} \geq 2 y_{h,k} \qquad \forall\, k \in K, h \in S, S \subseteq V^* \tag{2.37}$$

$$x_{i,j,k} \in \{0,1\} \qquad \forall\, i \in V, j \in V, k \in K \tag{2.38}$$

$$y_{i,k} \in \{0,1\} \qquad \forall\, i \in V, k \in K \tag{2.39}$$

$$z_{i,j,m,k} \in \{0,1\} \qquad \forall\, i \in V, j \in V, m \in V, k \in K \tag{2.40}$$

$$\tag{2.41}$$

The formulation is nearly identical, except for two changes. We have eliminated constraints (2.12) from the formulation in Sec. 2.2 since we no longer require particular sites to be visited. Additionally, we have changed the set notation of constraints (2.9) in Sec. 2.2 from "for all $i \in V \backslash T$" to "for all $i \in V^*$," as shown in (2.27) in the formulation shown in Sec. 2.3. This change now states that each site can be visited at most once by the fleet; that is, it does not need to be visited, but if it is, it can only be by one vehicle one time.

Chapter 3

Heuristics

3.1 Modified Sweep Heuristic

In this section we discuss an extended sweep heuristic adopted from the *modified sweep heuristic* in [16]. In particular, this algorithm constructs feasible solutions to the multi-vehicle covering tour problem, dispatching at most $|K|$ vehicles by utilizing the central concepts of the classical sweep algorithm from [14].

3.1.1 Applying the Sweep Algorithm to Formulation I (Sec. 2.2)

The algorithm is applied to the vertices of $T \cup W$. Note that Step 2 and Step 3 were presented by [16] and modified accordingly. The heuristic can be described as follows.

Step 1: Determine the angles ϕ_i of each vertex $v_i \in (T \cup W)\backslash\{v_0\}$ relative to the depot v_0, for $-\pi \leq \phi_i \leq \pi$.

Step 2: Determine the vertex v of $(T \cup W)\backslash\{v_0\}$ having the smallest angle ϕ and consider the half-line having an extremity at the depot v_0 and passing through v. If multiple vertices have the same angle, select the vertex with the smallest ordinate. Relabel all vertices $v_i \in (T \cup W)\backslash\{v_0\}$ in increasing order of their angle ϕ_i.

Step 3: Starting with the half-line v_0v, rotate a radius having a fixed point at v_0 in the counter-clockwise direction until a vertex $v_k \in (T \cup W)\backslash\{v_0\}$ is reached.

a) If $v_h \in T$, include it in the current route if this creates no infeasibility (i.e., the number of vertices of V in the current route does not exceed p; the length of the current route, including $c_{0,h}$, does not exceed q; and the energy used, including $e_{(h-1),h,0}$, does not exceed E), and continue the sweeping process. If v_h cannot feasibly be included into the current route, complete the current route by linking its last vertex to v_0, initialize a new route starting from v_0 to v_h, and continue the sweeping process.

b) If $v_h \in W$, determine the vertex $v_k \in \{v_i \mid c_{i,h} \leq r\}$ covering the largest number of vertices of W and include it in the current route if this is feasible.

   i If there are multiple vertices with the same sized covering set, choose the vertex $v_k \in T$.

   ii If there are no vertices in T that have the largest covering set, then choose $v_k \in V \backslash T$ without prejudice.

   Otherwise, complete the current route by linking its last vertex to v_0, initialize a current route starting from v_0 to v_h, and continue the sweeping process. If there is no $v_k \in \{v_i \mid c_{i,h} \leq r\}$, stop; there is no feasible solution.

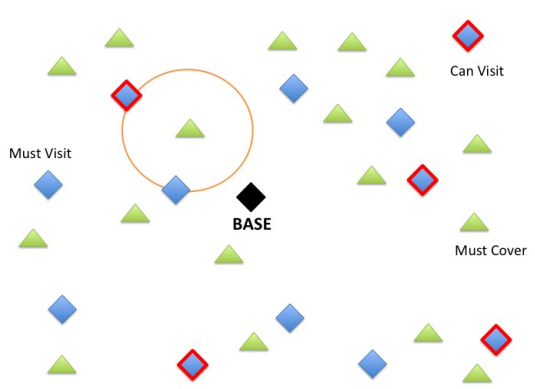
Step 4: If all vertices $v_t \in T \backslash \{v_0\}$ are routed and all vertices $v_w \in W$ are covered, stop. If the number of routes does not exceed $|K|$, the algorithm has yielded a feasible solution. If the number of routes exceeds $|K|$, the solution is not feasible.

The algorithm can be restarted from the vertex whose angle is the next smallest to generate a feasible solution. Additionally, if desired, an improvement heuristic can be applied to each tour to generate better results. Figure 3.1 provides a series of illustrations to demonstrate the sweep heuristic.

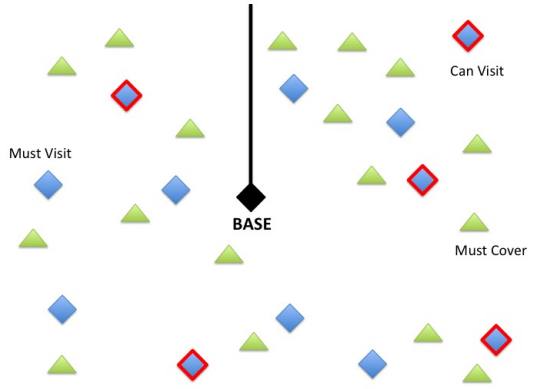
3.1.2 Applying the Sweep Algorithm to Formulation II (Sec. 2.3)

The algorithm is applied to the vertices of W. Note that Step 2 and Step 3 were presented by [16] and modified accordingly. The heuristic can be described as follows.

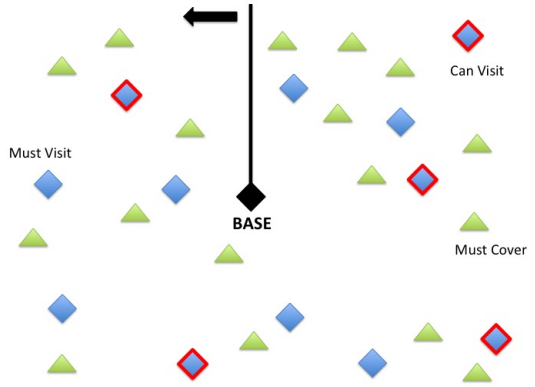
Step 1: Determine the angles ϕ_i of each vertex $v_i \in W$ relative to the depot v_0, for $-\pi \leq \phi_i \leq \pi$.



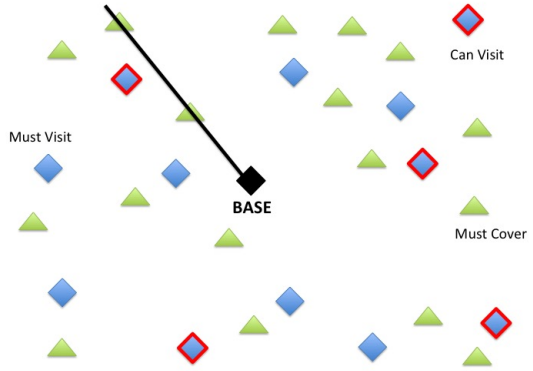
(a) Example site layout with coverage radius r shown



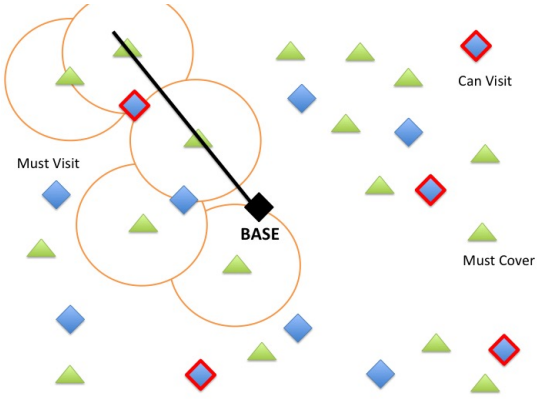
(b) Create line to sweep



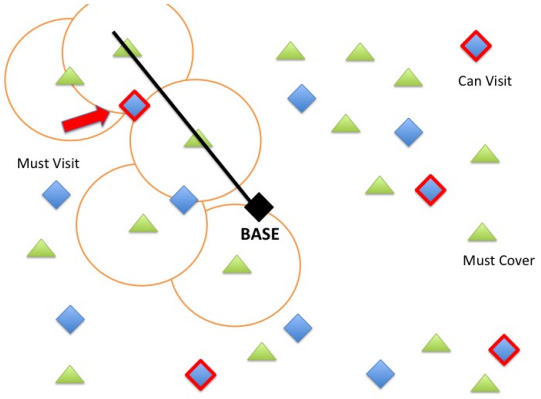
(c) Choose direction to sweep



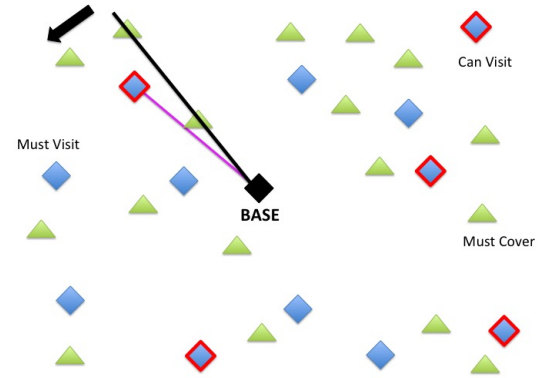
(d) Intersect point in W



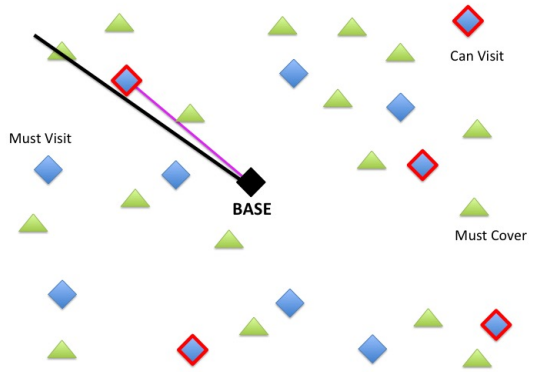
(e) Find the point in its covering set that covers the most points



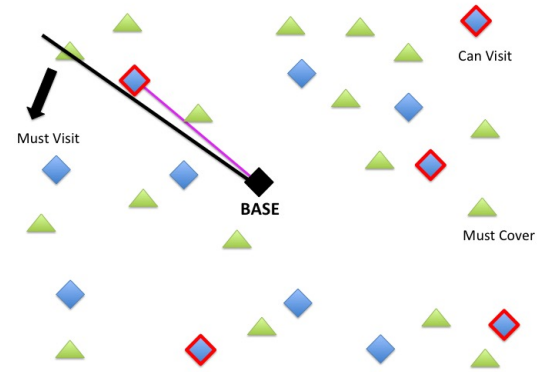
(f) Choose point in set that covers most points in W



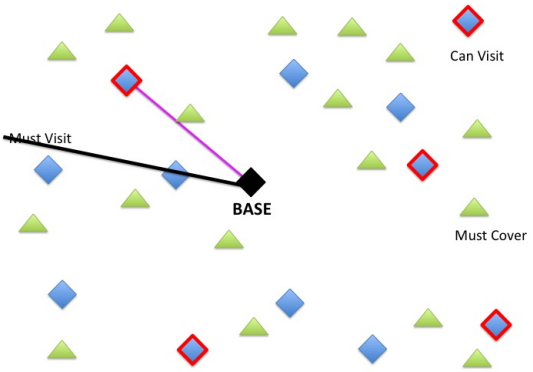
(g) Add point to path (if feasible) and continue sweeping



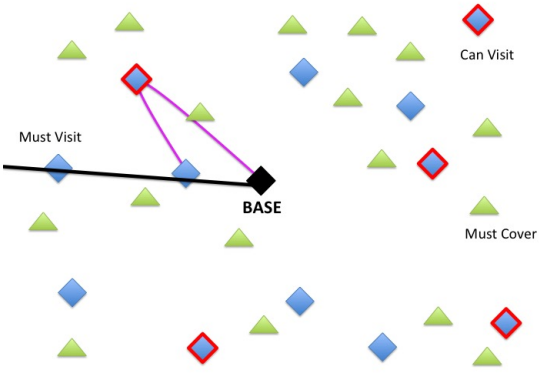
(h) We will intersect points that are already on the route or covered



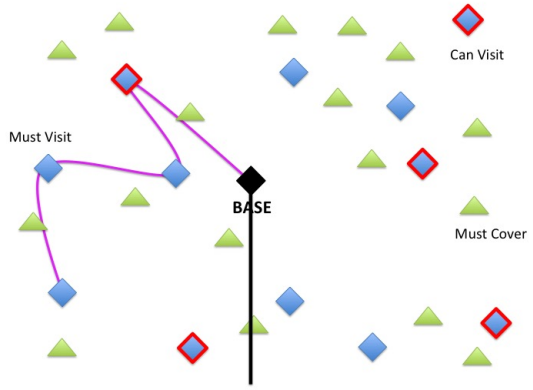
(i) Keep sweeping until intersect new point



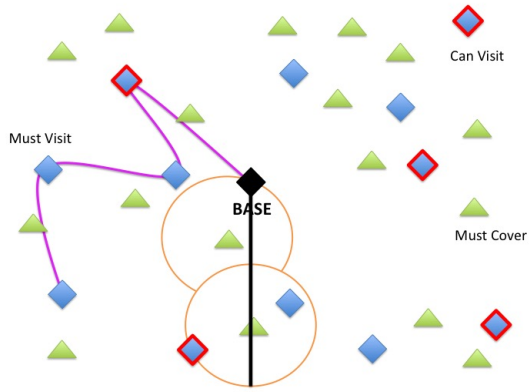
(j) Intersect point in T



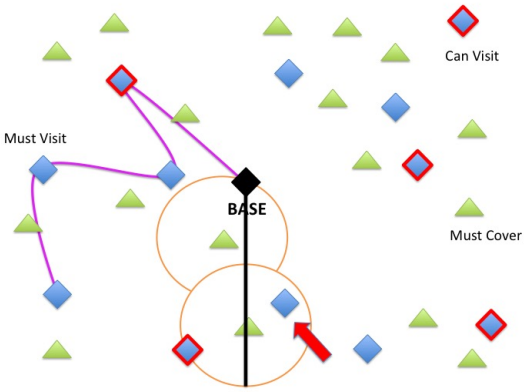
(k) Add to route (if feasible) and continue to sweep



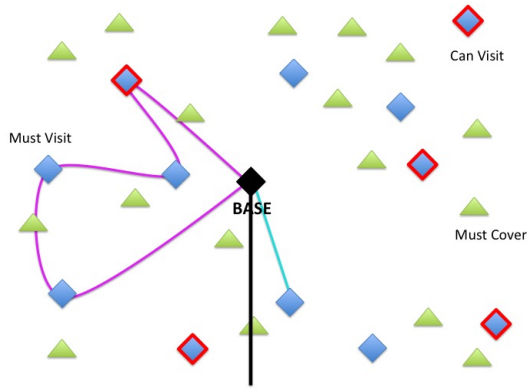
(l) After a few more iterations...



(m) Choose point in set that covers most points in W



(n) Add to path (if feasible)



(o) Not feasible, so end current route and start new one

Figure 3.1: Schematic of sweep heuristic for Formulation I

Step 2: Determine the vertex v of W having the smallest angle ϕ and consider the half-line having an extremity at the depot v_0 and passing through v. If multiple vertices have the same angle, select the vertex with the smallest ordinate. Relabel all vertices $v_i \in W$ in increasing order of their angle ϕ_i.

Step 3: Starting with the half-line v_0v, rotate a radius having a fixed point at v_0 in the counter-clockwise direction until a vertex $v_k \in W$ is reached.

a) If $v_h \in W$, determine the vertex $v_k \in \{v_i \mid c_{i,h} \leq r\}$ covering the largest number of vertices of W and include it in the current route if this it creates no infeasibility (i.e., the number of vertices of V in the current route does not exceed p; the length of the current route, including $c_{0,h}$, does not exceed q; and the energy used, including $e_{(h-1),h,0}$, does not exceed E), and continue the sweeping process. Otherwise, complete the current route by linking its last vertex to v_0, initialize a current route starting from v_0 to v_k, and continue the sweeping process. If there is no $v_k \in \{v_i \mid c_{i,h} \leq r\}$, stop; there is no feasible solution.

Step 4: If all vertices $v_w \in W$ are covered, stop. If the number of routes does not exceed $|K|$, the algorithm has yielded a feasible solution. If the number of routes exceeds $|K|$, the solution is not feasible.

The algorithm can be restarted from the vertex whose angle is the next smallest to generate a feasible solution. Additionally, if desired, an improvement heuristic can be applied to each tour to generate better results.

Notice that the two versions differ in that the former (3.1.1) applies to $T \cup W$, while the latter (3.1.2) applies to W.

3.2 Modified Savings Heuristic

In this section we discuss a modified savings heuristic adapted from the *modified savings heuristic* designed by [16]. In particular, this algorithm constructs feasible solutions to the multi-vehicle

covering tour problem, dispatching at most $|K|$ vehicles by utilizing the central concepts of the classical savings algorithm from [2].

3.2.1 Applying the Savings Algorithm to Formulation I (Sec. 2.2)

Note that Step 2 and Step 3 were presented by [16] and modified accordingly. The heuristic can be described as follows.

Step 1: Check to see if every vertex of W is covered by at least one vertex in V. If not, stop. Otherwise, continue.

Step 2: Construct a VRP solution on the graph induced by V by means of the parallel version of the Clarke and Wright algorithm, i.e., vehicle routes are gradually augmented according to the largest savings, while ensuring the augmentation does not create any infeasibility (i.e., the number of vertices of V in the current route does not exceed p; the length of the current route, including $c_{0,h}$, does not exceed q; and the energy used, including $e_{(h-1),h,0}$, does not exceed E).

Step 3: Compute for each vertex of V included in a vehicle route the savings obtained by removing it from the solution and reconnecting its predecessor and successor by an edge on the route. Sort these vertices in a list by decreasing order of savings. Consider each vertex v_k of the list and remove it if $v_k \notin T$ or this does not cause a vertex of W to be uncovered.

Step 4: Compute for each vertex of V included in a vehicle route the savings obtained by removing it from the solution, reconnecting its predecessor and successor by an edge on the route, and then adding the vertex to any other point in the current solution. Sort the destination for the vertex by decreasing order of savings. Make the change that has the greatest savings as long as this does not cause a vertex of W to be uncovered, nor create any infeasibility.

Step 5: Repeat *Step 3* for each vertex of $V \backslash \overline{V}$, where $\overline{V}$ is the set of all vertices displaced by *Step 3*. When there is no more positive savings generated by rearranging the routes, stop.

Step 6: If the total number of routes does not exceed $|K|$, the algorithm has yielded a feasible solution. Otherwise, no feasible solution can be find from this algorithm.

3.2.2 Applying the Savings Algorithm to Formulation II (Sec. 2.3)

Note that Step 2 and Step 3 were presented by [16] and modified accordingly. The heuristic can be described as follows.

Step 1: Check to see if every vertex of W is covered by at least one vertex in V. If not, stop. Otherwise, continue.

Step 2: Construct a VRP solution on the graph induced by V by means of the parallel version of the Clarke and Wright algorithm, i.e., vehicle routes are gradually augmented according to the largest savings, while ensuring the augmentation does not create any infeasibility (i.e., the number of vertices of V in the current route does not exceed p; the length of the current route, including $c_{0,h}$ does not exceed q; and the energy used, including $e_{(h-1),h,0}$ does not exceed E).

Step 3: Compute for each vertex of V included in a vehicle route the savings obtained by removing it from the solution and reconnecting its predecessor and successor by an edge on the route. Sort these vertices in a list by decreasing order of savings. Consider each vertex of the list and remove it if this does not cause a vertex of W to be uncovered.

Step 4: Compute for each vertex of V included in a vehicle route the savings obtained by removing it from the solution, reconnecting its predecessor and successor by an edge on the route, and then adding the vertex to any other point in the current solution. Sort the destination for the vertex by decreasing order of savings. Make the change that has the greatest savings as long as this does not cause a vertex of W to be uncovered, nor create any infeasibility.

Step 5: Repeat *Step 3* for each vertex of $V \backslash \overline{V}$, where $\overline{V}$ is the set of all vertices displaced by *Step 3*. When there is no more positive savings generated by rearranging the routes, stop.

Step 6: If the total number of routes does not exceed $|K|$, the algorithm has yielded a feasible solution. Otherwise, no feasible solution can be find from this algorithm.

The difference between the two algorithms is that the latter (3.2.2) is free to remove and reorder any site from any route it deems appropriate, while the former is forced to maintain certain sites on the routes and is limited to what sites can be present on a given route.

Chapter 4

Velocity Routing

In this chapter we introduce a model to determine the velocity of each vehicle between sites. The objective of the model is to minimize a weighted sum of the completion time of the locations to be visited.

4.1 Determining Vehicle Velocity for a Known Route

We assume that the unmanned surface vehicle (USV) has a predetermined route consisting of locations in the set V, including the depot, $\{0\}$. Further, we assume that the USV travels at a constant velocity along the arc between sites i and j and can instantaneously change its velocity for the subsequent arc. The distance between and two nodes is assumed to be known and the wave activity is assumed to be negligible.

Since we have a constant velocity, we can then express the time it takes for vehicle k to travel between points i and j, $t_{i,j,k}$, in terms of the distance between the two points, $c_{i,j}$, and the velocity of vehicle k between those points, $v_{i,j,k}$. More specifically, we have

$$t_{i,j,k} = \frac{c_{i,j}}{v_{i,j,k}}. \tag{4.1}$$

Moreover, since the USV has a finite battery life, we need to ensure that it does not deplete its entire energy reserve before it returns to the depot. The amount of energy needed to go from

i to j is dependent upon the power used. Specifically, we have

$$e_{i,j} = P_{i,j} t_{i,j,k}, \tag{4.2}$$

where $P_{i,j}$ is the power used. We assume

$$P_{i,j} = l_3 v_{i,j,k}^2. \tag{4.3}$$

for $l_3 \in \mathbb{R}$. It follows from Eqs. (4.1)-(4.3) that

$$\begin{aligned} e_{i,j} &= P_{i,j} t_{i,j,k} \\ &= l_3 v_{i,j,k}^2 \frac{c_{i,j}}{v_{i,j,k}} \\ &= l_3 v_{i,j,k} c_{i,j}. \end{aligned} \tag{4.4}$$

Note that the power used is roughly proportional to the cube of the velocity [15]. However, we opted to model it as proportional to the square of the velocity since it makes the energy linear in the velocity in (4.4).

Additionally, the angle of the turns made along the route affects the overall energy used, since it affects the velocity at which the vehicle can travel when going from site i to site j if site m is the subsequent stop. Thus, we say the energy lost by making a turn of angle $\theta_{i,j,m}$ when traveling from i to j with m as the subsequent stop is

$$l_4(\pi - \theta_{i,j,m}), \tag{4.5}$$

where $l_4 \in \mathbb{R}$. This is illustrated in Fig. 2.2. Thus, the total energy used when traveling from i to j with m as the subsequent stop is

$$e_{i,j,m} = l_3(v_{i,j,k} c_{i,j} + v_{j,m,k} c_{j,m}) + l_4(\pi - \theta_{i,j,m}). \tag{4.6}$$

Since we already know the route to be taken, it is our objective to determine the velocities of each arc while minimizing the time needed to complete the route. Additionally, each site $w \in W$

has a specific "importance weight", and when that weight, α_w, is high we want the USV to travel more slowly. Since each arc (i, j) may cover multiple sites w, we define the importance weight for each arc as

$$\alpha_{i,j} = \frac{\sum_{w \in W} \alpha_w h_{i,j,w}}{\sum_{w \in W} h_{i,j,w}}. \tag{4.7}$$

Hence, we define the set $\overline{V}_k = \{(i, j) \mid x_{i,j,k} = 1\}$ and $\tilde{V}_k = \{(i, j, m) \mid z_{i,j,,m,k} = 1\}$. Using this notation, our formulation is:

$$\text{minimize} \quad \sum_{k \in K} \sum_{(i,j) \in \overline{V}_k} \frac{c_{i,j} \alpha_{i,j}}{v_{i,j,k}} \tag{4.8}$$

$$\sum_{(i,j,m) \in \tilde{V}_k} e_{i,j,m} \leq E \qquad \forall\ k \in K \tag{4.9}$$

$$v_{i,j,k} \leq \tau \qquad \forall\ (i, j) \in \overline{V}_k,\ k \in K \tag{4.10}$$

$$v_{i,j,k} \geq 0 \qquad \forall\ (i, j) \in \overline{V}_k,\ k \in K \tag{4.11}$$

The objective (4.8) is to minimize a weighted sum of the completion time of the locations to be visited. Constraints (4.9) ensure that each vehicles does not deplete its energy reserves. Constraints (4.10) prevents each vehicles from exceeding a maximum velocity τ along any arc, and constraints (4.11) are the standard non-negativity conditions. Note that we can rewrite Eq. (4.9) as

$$\sum_{(i,j,m) \in \tilde{V}_k} l_3(v_{i,j,k} c_{i,j} + v_{j,m,k} c_{j,m}) + l_4(\pi - \theta_{i,j,m}) \leq E \quad \forall\ k \in K, \tag{4.12}$$

where $\theta_{i,j,m}$ is defined in Eq. (2.3).

Note that we have a nonlinear, convex objective function and linear, convex constraint. Thus, solving this problem will yield a globally optimal solution.

4.2 The Multi-Vehicle Covering Tour Problem with Velocities

Coastal surveillance is a means for patrolling the coastline to detect, deter, and respond to a range of threats; it can serve a range of objectives. Whether it be bomb detection, crime prevention, locating enemy combatants, or emergency response, coastal surveillance is a tractable and benefi-

cial system. Using a fleet of unmanned surface vehicles (USVs), we are tasked with constructing efficient routes to surveil specified locations. Each USV, though, operates on battery power, and thus must return to the central depot prior to draining its energy reserves. However, some sites are more of a security threat than others; hence, we seek to not only covers these sites but to optimize the time spent surveilling along a given route by focusing more time on greater threats.

We maintain the same notation as in Sec. 2.3, except for a few adjustments. Let Q be the maximum allowable total distance traveled for the fleet. Additionally, we define the energy used when traveling from i to j, with m as the next stop, according to Eq. (4.12). Furthermore, we define the importance weight of each arc as in Eq. (4.7). Moreover, we introduce two new decision variables both representing velocity:

$$v_{i,j,k} = \begin{cases} \text{the velocity of vehicle } k \text{ from site } i \text{ to site } j, \text{if vehicle } k \text{ travels from site } i \text{ to site } j \\ M, \quad \text{otherwise;} \end{cases} \tag{4.13}$$

and,

$$g_{i,j,k} = \begin{cases} \text{the velocity of vehicle } k \text{ from site } i \text{ to site } j, \text{if vehicle } k \text{ travels from site } i \text{ to site } j \\ 0, \quad \text{otherwise.} \end{cases} \tag{4.14}$$

Thus, the formulation is as follows:

$$\text{minimize} \quad \sum_{k\in K}\sum_{i\in V}\sum_{j\in V}\frac{c_{i,j}\alpha_{i,j}}{v_{i,j,k}} \tag{4.15}$$

$$\sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{i,j}x_{i,j,k} \le Q \tag{4.16}$$

$$\sum_{k\in K}\sum_{i\in V}\sum_{j\in V\setminus\{i\}} h_{i,j,w}\ x_{i,j,k} \ge 1 \qquad \forall\ w \in W \tag{4.17}$$

$$\sum_{k\in K} y_{i,k} \le 1 \qquad \forall\ i \in V^* \tag{4.18}$$

$$\sum_{i\in V}\sum_{j\in V\setminus\{i\}} c_{i,j}x_{i,j,k} \le q \qquad \forall\ k \in K \tag{4.19}$$

$$\sum_{i \in V^*} y_{i,k} \leq p \qquad \forall\ k \in K \tag{4.20}$$

$$y_{0,k} \leq 1 \qquad \forall\ k \in K \tag{4.21}$$

$$\sum_{j \in V \setminus \{i\}} x_{i,j,k} = y_{i,k} \qquad \forall\ i \in V, k \in K \tag{4.22}$$

$$\sum_{j \in V \setminus \{i\}} x_{j,i,k} = y_{i,k} \qquad \forall\ i \in V, k \in K \tag{4.23}$$

$$v_{i,j,k} \geq x_{i,j,k} + M(1 - x_{i,j,k}) \qquad \forall\ i \in V,\ j \in V,\ k \in K \tag{4.24}$$

$$v_{i,j,k} \leq \tau x_{i,j,k} + M(1 - x_{i,j,k}) \qquad \forall\ i \in V,\ j \in V,\ k \in K \tag{4.25}$$

$$g_{i,j,k} \geq x_{i,j,k} \qquad \forall\ i \in V,\ j \in V,\ k \in K \tag{4.26}$$

$$g_{i,j,k} \leq \tau x_{i,j,k} \qquad \forall\ i \in V,\ j \in V,\ k \in K \tag{4.27}$$

$$v_{i,j,k} - g_{i,j,k} \leq M(1 - x_{i,j,k}) \qquad \forall\ i \in V,\ j \in V,\ k \in K \tag{4.28}$$

$$v_{i,j,k} - g_{i,j,k} \geq -M(1 - x_{i,j,k}) \qquad \forall\ i \in V,\ j \in V,\ k \in K \tag{4.29}$$

$$\sum_{i \in V} \sum_{j \in V^* \setminus \{i\}} \sum_{m \in V \setminus \{j\}} l_3(g_{i,j,k}c_{i,j} + g_{j,m,k}c_{j,m}) + l_4(\pi - \theta_{i,j,m})z_{i,j,m,k} \leq E \quad \forall\ k \in K \tag{4.30}$$

$$z_{i,j,m,k} \leq x_{i,j,k} \qquad \forall\ i \in V, j \in V^* \setminus \{i\}, m \in V \setminus \{j\}, k \in K \tag{4.31}$$

$$z_{i,j,m,k} \leq x_{j,m,k} \qquad \forall\ i \in V, j \in V^* \setminus \{i\}, m \in V \setminus \{j\}, k \in K \tag{4.32}$$

$$z_{i,j,m,k} \geq x_{i,j,k} + x_{j,m,k} - 1 \qquad \forall\ i \in V, j \in V^* \setminus \{i\}, m \in V \setminus \{j\}, k \in K \tag{4.33}$$

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{i,j,k} + \sum_{i \in S} \sum_{j \in V \setminus S} x_{j,i,k} \geq 2y_{h,k} \qquad \forall\ k \in K, S \subseteq V^*, h \in S \tag{4.34}$$

$$x_{i,j,k} \in \{0, 1\} \qquad \forall\ i \in V, j \in V, k \in K \tag{4.35}$$

$$y_{i,k} \in \{0, 1\} \qquad \forall\ i \in V, k \in K \tag{4.36}$$

$$z_{i,j,m,k} \in \{0, 1\} \qquad \forall\ i \in V, j \in V, m \in V, k \in K \tag{4.37}$$

The objective (4.15) is to minimize the total time traveling for the entire fleet while determining the velocity for each arc based upon the arc's importance. Constraints (4.16) limit the fleet's total distance to be less than Q. Constraints (4.17) ensure that each site in W is covered at least once. Constraints (4.18) dictate that each site in V^* is visited at most once. Constraints (4.19) limit the length of each route. Constraints (4.20) specify that each vehicle can visit at most p sites

(excluding the base). Constraints (4.21) make sure that each vehicle goes to the base if it is deployed. Constraints (4.22) ensure that there is only one route leaving i for vehicle k if vehicle k visits point i. Constraints (4.23) ensure that there is only one route entering i for vehicle k if vehicle k visits point i. Constraints (4.24) and (4.25) ensure that $1 \leq v_{i,j,k} \leq \tau \ \forall i \in V,\ j \in V,\ k \in K$ such that $x_{i,j,k} = 1$, and $v_{i,j,k} = M$ otherwise. Constraints (4.26) and (4.27) ensure that $1 \leq g_{i,j,k} \leq \tau \ \forall i \in V,\ j \in V,\ k \in K$ such that $x_{i,j,k} = 1$, and $g_{i,j,k} = 0$ otherwise. Constraints (4.28) and (4.29) impose the restriction that $v_{i,j,k}$ and $g_{i,j,k}$, the two variables representing the velocity of vehicle k along the arc (i, j), are equal when vehicle k traverses arc (i, j). Constraints (4.30) are the energy constraint, making sure that vehicle k's route does not exceed the battery capacity. If vehicle k traverses arc (i, j), then energy is used and accounted for; otherwise, it is not. Constraints (4.31)-(4.33) ensure that $z_{i,j,m,k} = 1$ if and only if vehicle k goes from $i \rightarrow j$ and then $j \rightarrow m$. Constraints (4.34) are the subtour elimination constraints. The remaining constraints correspond to the standard integrality conditions for the decision variables.

Chapter 5

Computational Results

5.1 Routing

In the section, we seek to compare the the efficiency of CPLEX and the heuristics outlined in Chapter 3 for the multi-vehicle covering tour problem introduced in Chapter 2. We generate four instances, with Euclidean distances, of varying sizes and parameter values, as shown in Table 5.1. Datasets are available from the author upon request.

For notational purposes, we define the optimality gap as the percentage difference between the objective value of the optimal solution and the objective value of the solution returned by the heuristic; that is, it is the percentage by which the heuristic's objective value is greater than that of the optimal solution as found by CPLEX. When comparing the solution times, moreover, a positive percentage indicates that the heuristic was slower when compared to CPLEX, while a

	4_0_8	**4_4_16**	**6_6_12**	**12_14_21**
$\lvert T^* \rvert$	4	4	6	12
$\lvert V^* \backslash T^* \rvert$	0	4	6	14
$\lvert W \rvert$	8	16	12	21
$\lvert K \rvert$	2	2	3	3
p	2	4	4	4
q	5	7	30	200
r	0.5	0.5	6	40
E	25	25	1000	300
τ	10	10	10	10

Table 5.1: Description of problem instances

Instance Name	Solution Method	Route	Elapsed Time (s)	Objective	Optimality Gap (%)	Time vs. CPLEX (%)
4_0_8	CPLEX	1-4-3-1 1-2-5-1	0.060	9.657	0.000	0.000
	Sweep	1-4-3-1 1-2-5-1	0.070	9.657	0.000	16.027
	Savings	1-3-2-1 1-5-4-1	0.288	9.657	0.000	380.500
4_4_16	CPLEX	1-8-3-2-5-1 1-9-4-1	5.660	10.065	0.000	0.000
	Sweep	1-4-8-3-7-1 1-2-6-5-1	0.072	10.333	2.666	-98.736
	Savings	1-3-2-5-1 1-8-4-9-1	0.419	10.065	0.000	-92.602
6_6_12	CPLEX	1-7-6-1 1-5-2-3-1 1-4-1	320.930	67.537	0.000	0.000
	Sweep	1-6-7-1 1-4-1 1-2-3-5-1	0.073	69.189	2.446	-99.977
	Savings	1-5-2-3-1 1-4-1 1-7-6-1	0.558	67.537	0.000	-99.826
12_14_21	CPLEX	Out of Memory Error				
	Sweep	1-12-6-9-11-1 1-10-13-4-7-1 1-2-8-3-5-1	0.086	438.999	—	—
	Savings	1-7-12-5-3-1 1-10-11-9-6-1 1-4-13-2-8-1	7.7446	390.069	—	—

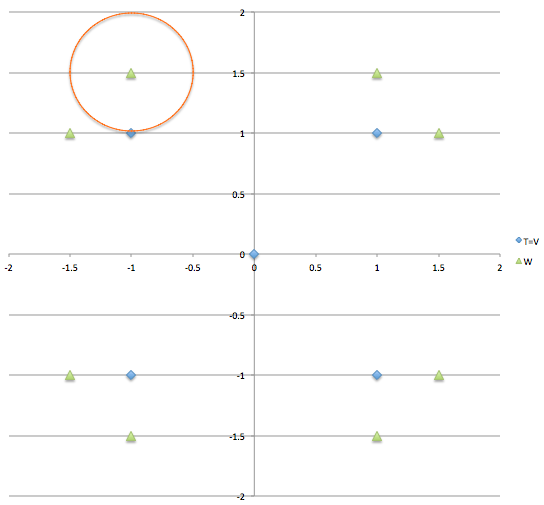
Table 5.2: Results for Formulation I

negative percentage indicates that it was faster.

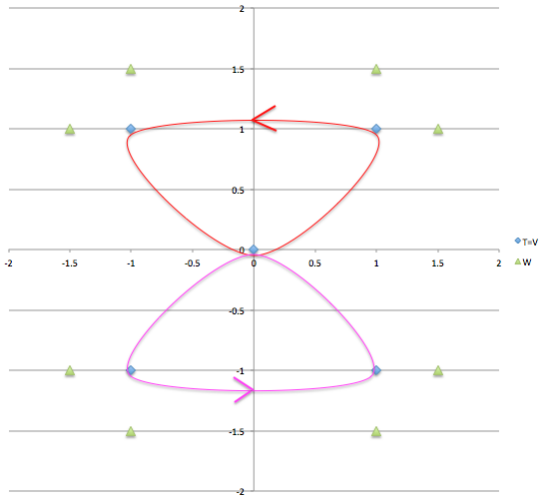
All tests were run on a Linux-Debian machine with 32GB RAM and 16 AMD Opteron^{TM} 6128 Magny-Cours 2.0 GHz processors. We used one processor and solved the model using CPLEX 12.6.1.0. The heuristics described in Chapter 3 were coded in MATLAB (R2014b (8.4.0.150421) 64-bit (glnxa64)).

5.1.1 Formulation I

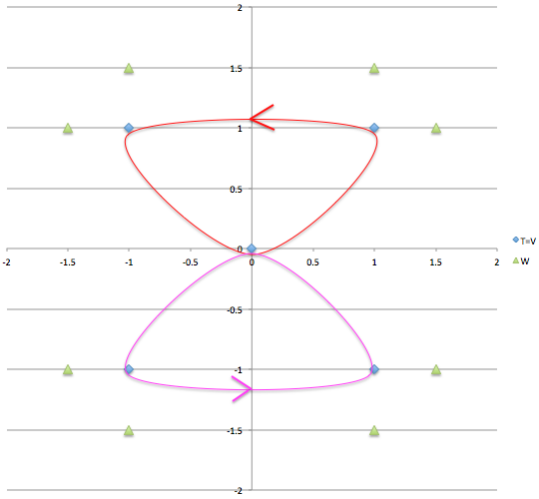
We focus our computational experiments on Formulation I as outlined in Sec. 2.2 and formulate the model as an integer program using AMPL. Table 5.2 summarizes the results.



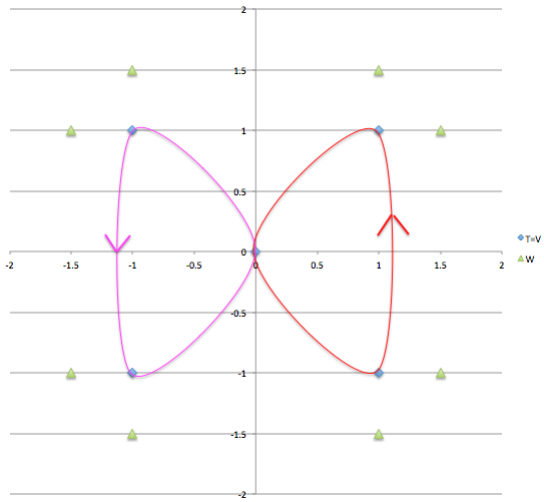
(a) Site layout with coverage radius



(b) CPLEX solution



(c) Modified sweep solution

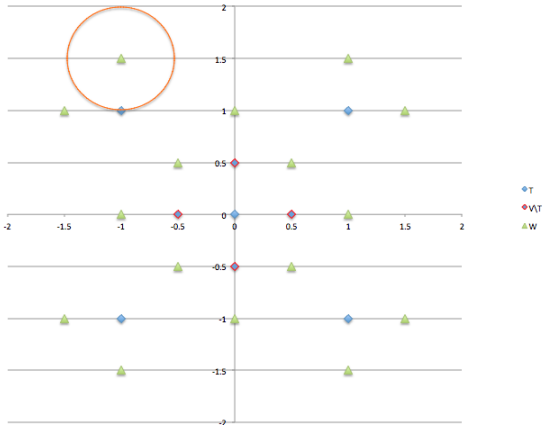


(d) Modified savings solution

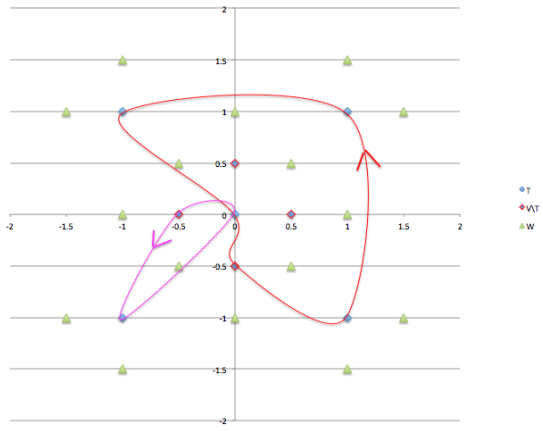
Figure 5.1: Site layout and optimal/heuristic solutions for Instance 4_0_8 for Formulation I.

As the instances grow in size, it takes longer for CPLEX to obtain an optimal solution. Notice that in the largest dataset, CPLEX failed to obtain a feasible solution because it exceeded the available memory. As the size of the problem increased, the time CPLEX needed to solve the given instance also increased.

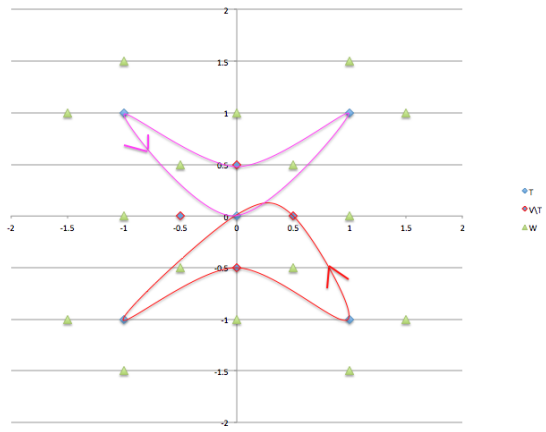
Furthermore, the sweep algorithm found a feasible solution in less time than the savings algorithm, though the solution was typically worse. That is, though the savings algorithm took longer to run, it attained the optimal objective value for all instances for which CPLEX did, though their optimal solutions may differ. The site layouts for each instance and their corresponding optimal/heuristic solutions are shown in Figs. 5.1, 5.2, 5.3, and 5.4.



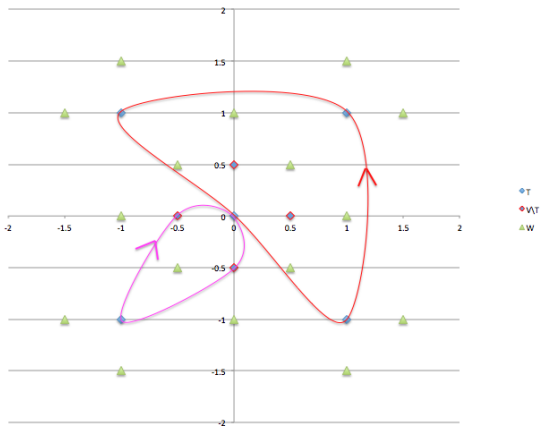
(a) Site layout with coverage radius



(b) CPLEX solution

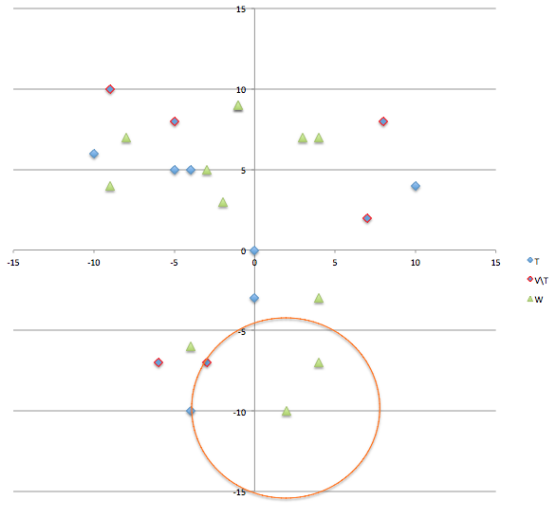


(c) Modified sweep solution

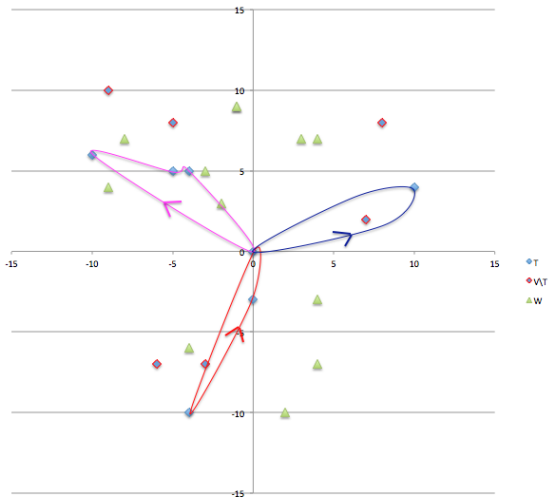


(d) Modified savings solution

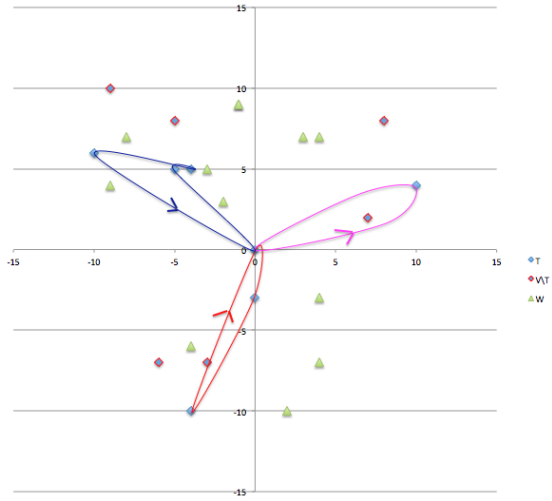
Figure 5.2: Site layout and optimal/heuristic solutions for Instance 4_4_16 for Formulation I.



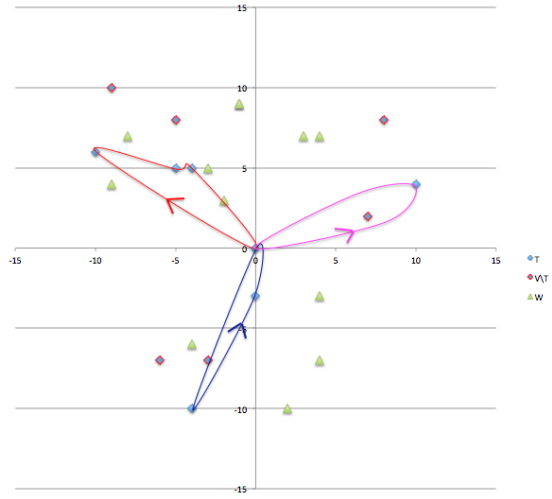
(a) Site layout with coverage radius



(b) CPLEX solution

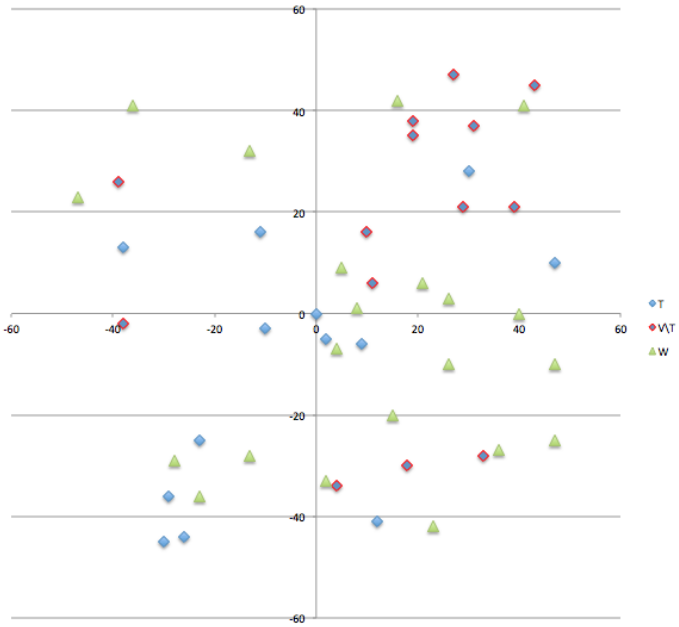


(c) Modified sweep solution

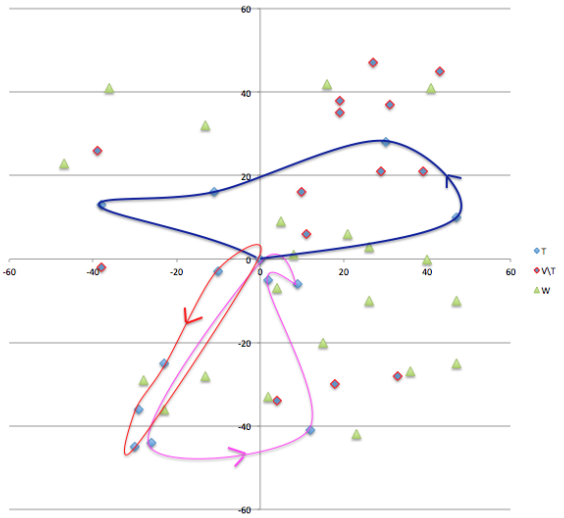


(d) Modified savings solution

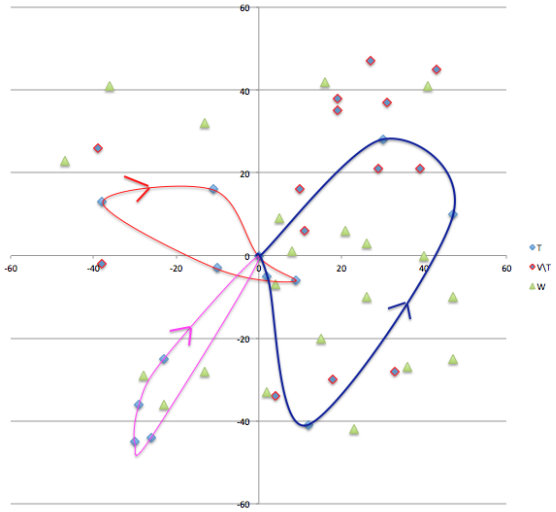
Figure 5.3: Site layout and optimal/heuristic solutions for Instance 6_6_12 for Formulation I.



(a) Site layout

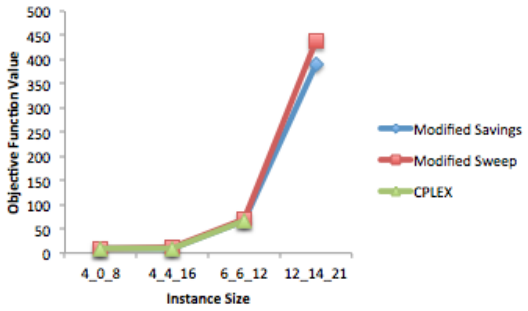


(b) Modified sweep solution

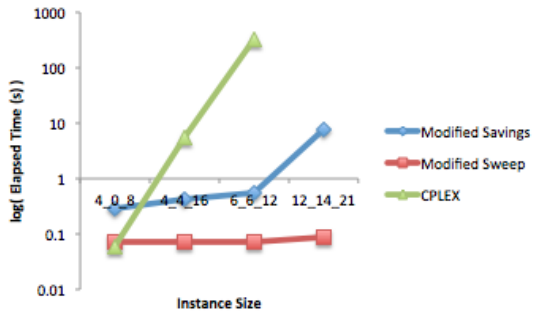


(c) Modified savings solution

Figure 5.4: Site layout and optimal/heuristic solutions for Instance 12_14_21 for Formulation I.



(a) Graph of objective function value vs. instance size for each heuristic.



(b) Graph of elapsed time (s) vs. instance size for each heuristic.

Figure 5.5: Comparison of CPLEX, Modified Savings, and Modified Sweep heuristics for Formulation I.

Figure 5.5 shows the objective function value and solution times for each method as the size of the instances grow. We see that as the size of the problem increases, both heuristics obtain a feasible solution faster than CPLEX (Fig. 5.5a). Moreover, we see that the modified sweep heuristic obtains a feasible solution the fastest for each instance. However, the difference between the objective function value of the modified sweep and CPLEX/modified savings increases as the problem size increases (Fig. 5.5b).

5.1.2 Formulation II

In this section, we focus our computational experiments on Formulation II as outlined in Sec. 2.3 and formulate the model as an integer program using AMPL. Table 5.3 -summarizes the results. We use the same instances in Table 5.1, but we set $V = V^* \cup T$ and eliminate the set T^*.

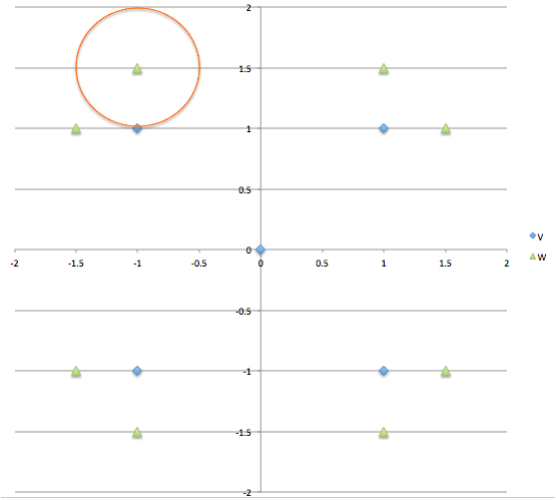
Similar to the previous formulation, as the instances grow in size, it takes longer for CPLEX to obtain an optimal solution. Notice that in the largest dataset, CPLEX failed to obtain a feasible solution since it exceeded the allotted memory.

Furthermore, the sweep algorithm found a feasible solution in less time than the savings algorithm, though the solution was typically worse. That is, though the savings algorithm took longer to run, it attained the optimal objective value for all instances for which CPLEX did, though their optimal solutions may differ. The site layouts for each instance and their corresponding optimal/heuristic solutions are shown in Figs. 5.6, 5.7, 5.8, and 5.9.

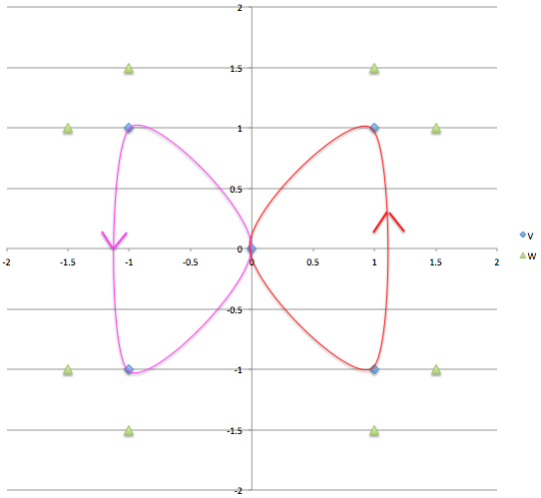
Figure 5.10 shows the objective function value and solution times for each method as the size

Instance Name	**Solution Method**	**Route**	**Elapsed Time (s)**	**Objective**	**Optimality Gap (%)**	**Time vs. CPLEX (%)**
4_0_8	CPLEX	1-3-2-1 1-5-4-1	0.050	9.657	0.000	0.000
	Sweep	1-4-3-1 1-2-5-1	0.185	9.657	0.000	270.000
	Savings	1-3-2-1 1-5-4-1	0.520	9.657	0.000	940.000
4_4_16	CPLEX	1-7-3-4-1 1-2-5-9-1	4.14	10.065	0.000	0.000
	Sweep	1-4-8-3-7-1 1-2-6-5-9-1	0.287	10.537	4.68	-93.068
	Savings	1-3-2-5-1 1-8-4-9-1	1.076	10.065	0.000	-74.010
6_6_12	CPLEX	1-3-10-1 1-9-6-1	12896.800	40.701	0.000	0.000
	Sweep	1-6-7-1 1-10-2-1	0.434	48.553	19.292	-99.997
	Savings	1-3-10-1 1-9-6-1	1.305	40.701	0.000	-99.990
12_14_21	CPLEX	Out of Memory Error				
	Sweep	1-4-7-21-23-1 1-3-1	2.278	142.4797	—	—
	Savings	1-21-1 1-5-22-1	12.576	214.860	—	—

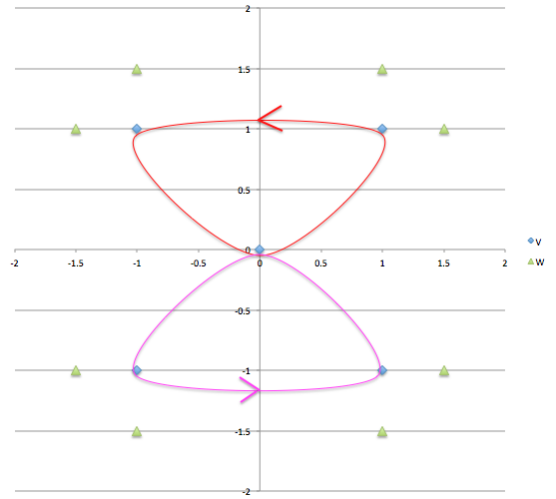
Table 5.3: Results for Formulation II



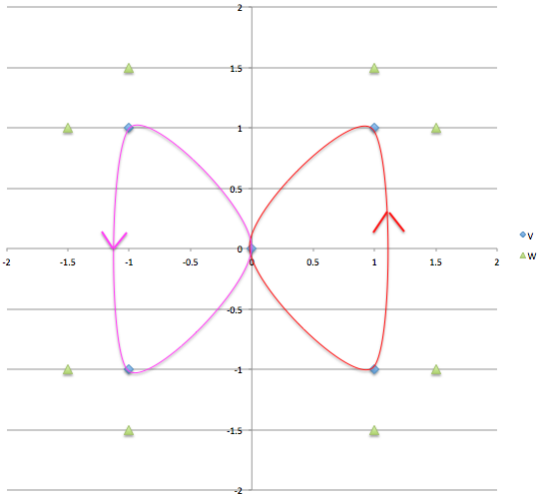
(a) Sit layout with coverage radius



(b) CPLEX solution

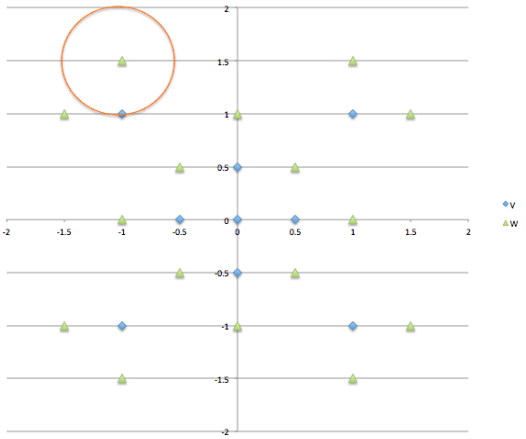


(c) Modified sweep solution

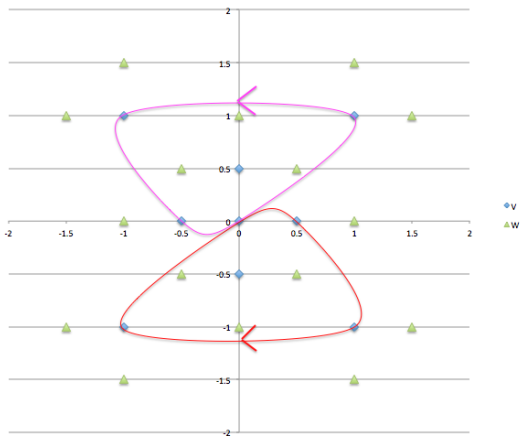


(d) Modified savings solution

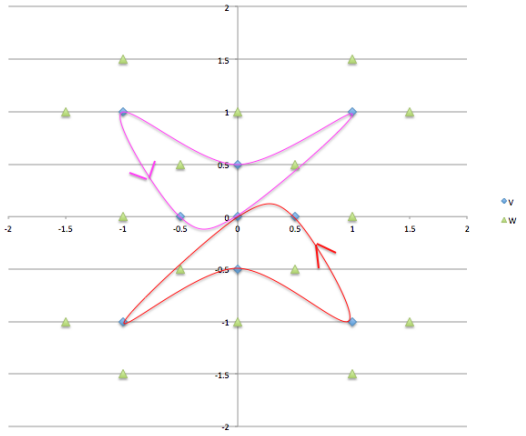
Figure 5.6: Site layout and optimal/heuristic solutions for Instance 4_0_8 for Formulation II.



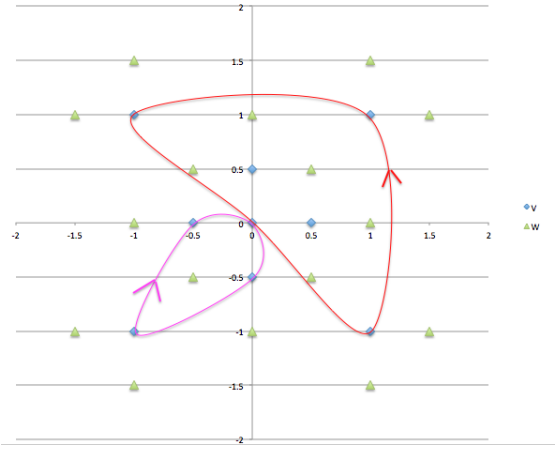
(a) Site layout with coverage radius



(b) CPLEX solution

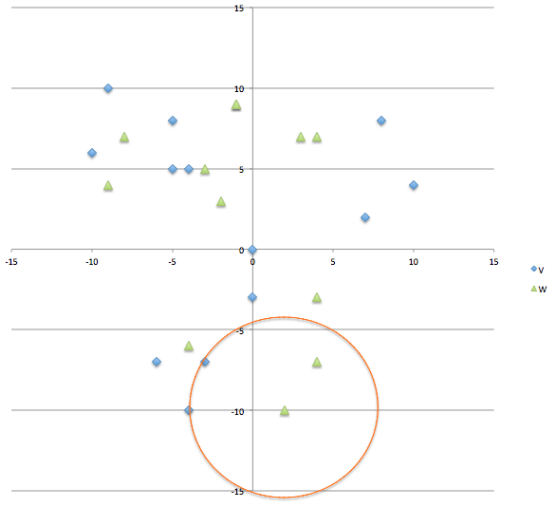


(c) Modified sweep solution

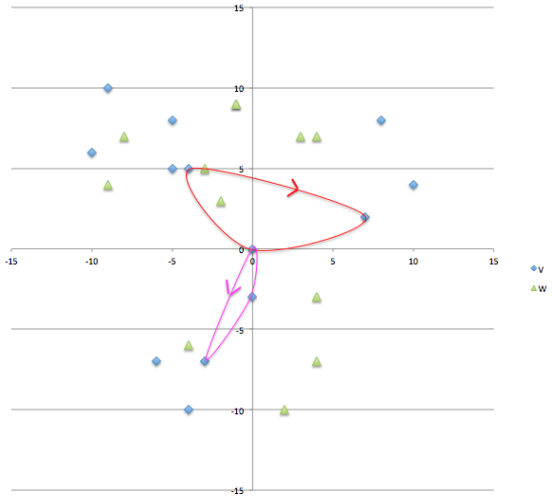


(d) Modified savings solution

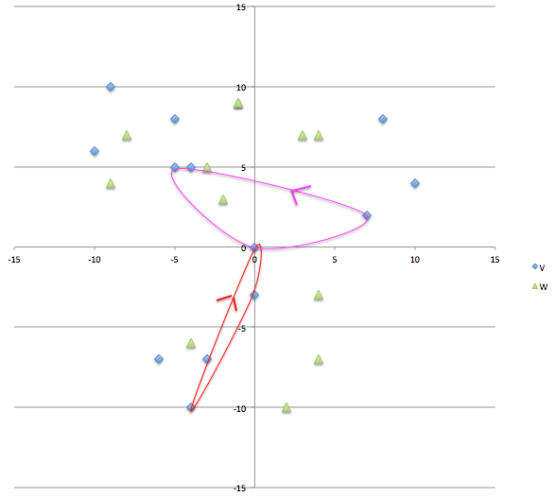
Figure 5.7: Site layout and optimal/heuristic solutions for Instance 4_4_16 for Formulation II.



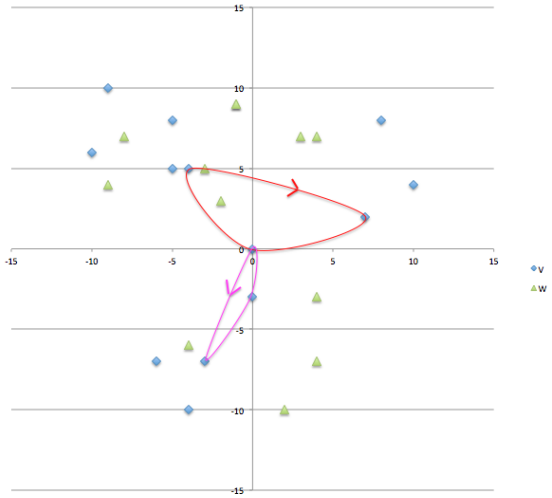
(a) Site layout with coverage radius



(b) CPLEX solution

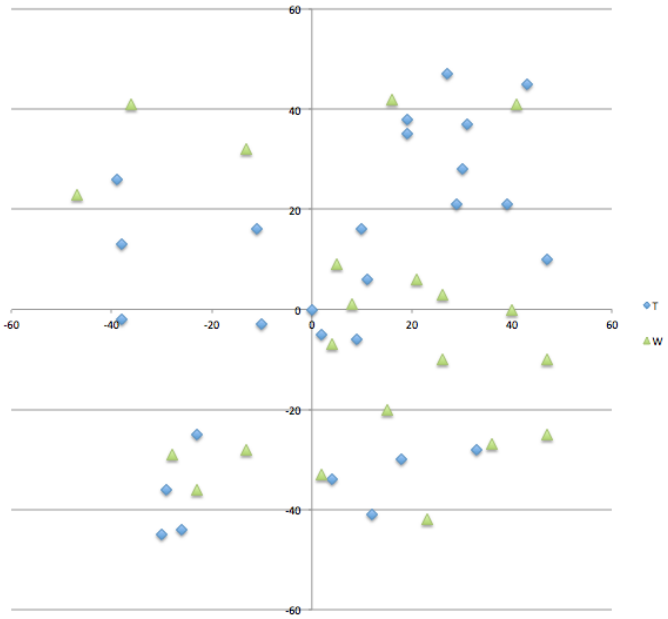


(c) Modified sweep solution

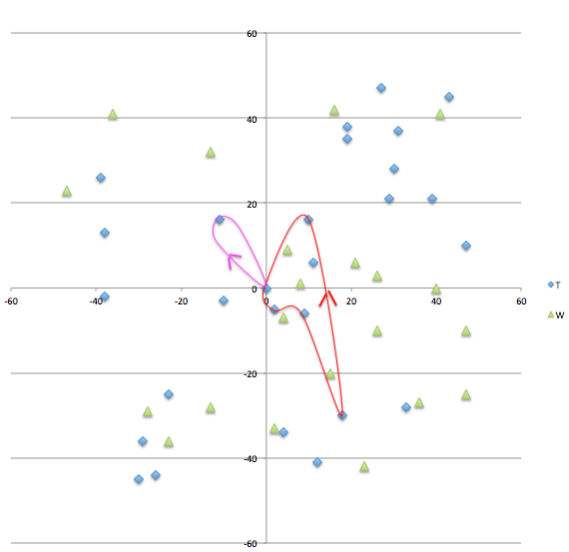


(d) Modified savings solution

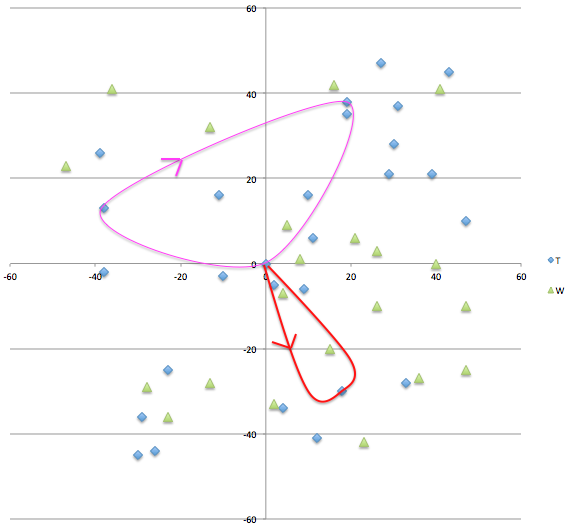
Figure 5.8: Site layout and optimal/heuristic solutions for Instance 6_6_12 for Formulation II.



(a) Site layout

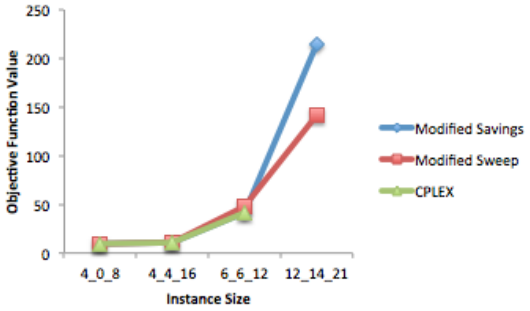


(b) Modified sweep solution

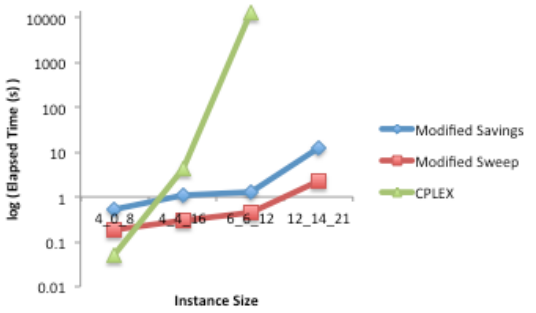


(c) Modified savings solution

Figure 5.9: Site layout and optimal/heuristic solutions for Instance 12_14_21 for Formulation II.



(a) Graph of objective function value vs. instance size for each heuristic.



(b) Graph of elapsed time (s) vs. instance size for each heuristic.

Figure 5.10: Comparison of CPLEX, Modified Savings and Modified Sweep heuristics for Formulation II.

of the instances grow. We see that as the size of the problem increases, both heuristics obtain a feasible solution faster than CPLEX (Fig. 5.10a). Moreover, we see that the modified sweep heuristic obtains a feasible solution the fastest for each instance. However, the difference between the objective function value of the modified sweep and CPLEX/modified savings increases as the problem size increases (Fig. 5.10b).

Furthermore, we note that it took longer for the heuristics in Sec. 3.1.2 and 3.2.2 to attain a solution than those in Sec. 3.1.1 and 3.2.1, though the instances were the same. We believe the reason stems from the presence of the set T. With the former heuristics, they must cycle through a larger set of possible solutions, whereas the latter has a more limited set due to the restriction placed upon the vehicles.

5.2 Routing with Velocity

All tests were run on a Linux-Debian machine with 32GB RAM and 16 AMD Opteron^{TM} 6128 Magny-Cours 2.0 GHz processors. Datasets are available from the author upon request.

We formulated the model in Sec. 4.2 for the multi-vehicle covering tour problem as a mixed integer, nonlinear program using AMPL and solved the problem with Bonmin 1.7.4 using Cbc 2.8.8 and Ipopt 3.11.7 and used one processor. The problem instances are shown in Fig. 5.4; they are similar to those in Fig. 5.1 with some modifications. In particular, we use the same set notation as that of Formulation II by setting $V = V^* \cup T$ and eliminating the set T^*.

The velocities for Instance 4_8, and the tour descriptions, are shown in Table 5.5. CPLEX

	4_8	**8_16**	**12_12**	**26_21**
$\|V^*\|$	4	8	12	26
$\|W\|$	8	16	12	21
$\|K\|$	2	2	3	3
p	2	4	4	4
q	5	7	30	200
r	0.5	0.5	6	40
E	50	150	1500	5000
τ	10	10.1	40.8	250
M	10^5	10^5	10^5	10^5

Table 5.4: Parameter modifications for velocity routing

Vehicle 1		**Vehicle 2**	
Arc	**Velocity (m/s)**	**Arc**	**Velocity (m/s)**
1-3	1.946	1-2	1.946
3-4	1.363	2-5	1.363
4-1	1.946	5-1	1.946

Table 5.5: Route and velocity results for Instance 4_8

obtained the optimal solution in 604.03 seconds. Note by introducing velocity as a decision variable, the run-time increases by a factor of roughly 10^4.

We map the optimal routes in Fig. 5.11, where both velocity and arc importance are illustrated. We can see that when the arc importance is high, the vehicles travel more slowly.

We did not test the latter two instances (12_12 and 26_21) since their run times were too long. Instance 8_16 terminated after two hours but was able to yield a feasible solution, as shown in Table 5.6. Due to our results from Sec. 5.3 we know the routing output is optimal, but cannot make that claim pertaining to the velocity values. Hence, we can only claim a feasible solution.

Vehicle 1		**Vehicle 2**	
Arc	Velocity (m/s)	Arc	Velocity (m/s)
1-6	3.907	1-8	3.907
6-5	2.074	8-3	2.074
5-4	1.641	3-2	1.641
4-1	2.555	2-1	2.555

Table 5.6: Route and velocity results for Instance 8_16

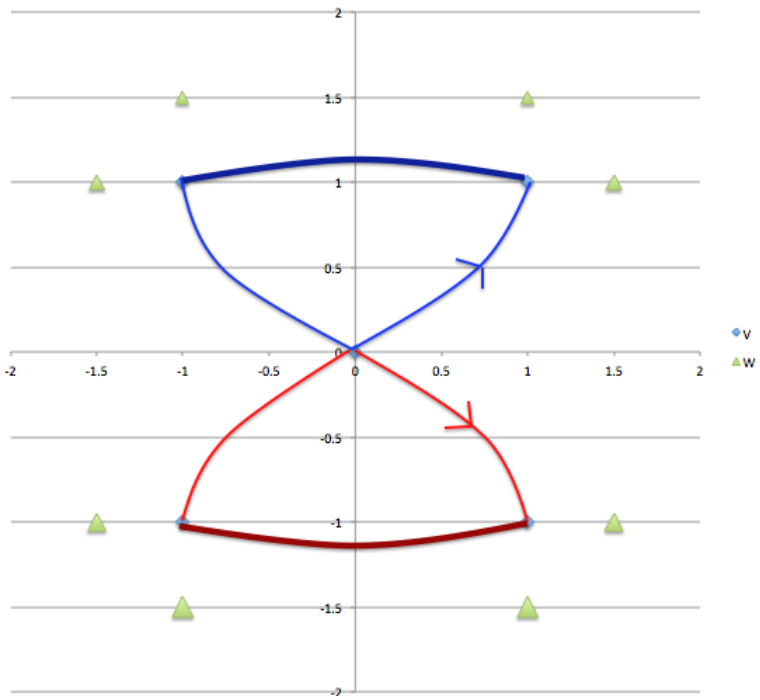


Figure 5.11: Map of vehicle routes for instance 4_8, where line thickness indicates arc importance and color intensity indicates velocity (the brighter the arc, the greater the velocity). For sites $w \in W$, marker size represents site importance.

Chapter 6

Conclusion

In this thesis, we formulated a model to route unmanned surface vehicles with the goal of minimizing total distance, subject to side constraints. In particular, we expanded upon the multi-vehicle covering tour problem from [16]. Specifically, we extended the definition of *covering* to account for a location being within a predetermined distance of any point on the tour, not just the stops. Furthermore, we introduced an additional constraint for the limited energy supply of the vehicles. Modifying two heuristics presented in [16], we implemented the algorithms in MATLAB and compared their efficiency to that of CPLEX, with which the integer program was solved to optimality. In doing so, we note that the heuristics attained feasible solutions faster than CPLEX on the larger instances; CPLEX, moreover, failed to solve some instances due to the size of the problem. It is also important to note that the savings algorithm presented in Sec 3.2 found the optimal solution in all instances for which CPLEX did.

We further developed our model by introducing velocity as a decision variable. Surveillance with unmanned surface vehicles in any locale is time sensitive. It is imperative to closely examine high valued areas, while a less thorough inspection is permitted for less vulnerable sites. As such, these unmanned surface vehicles must travel much slower while inspecting sites with a greater security risks. In doing so, the sensors and cameras attached to the vessels can closely examine the site. With this in mind, we sought to not only route these vehicles via waypoints to surveil locales, but also to determine the speed they must travel on each part of the route.

We implemented this larger model using a nonlinear optimization solver, and found that the

amount of time required to obtain optimality significantly increases from the pervious models. Introducing additional decision variables, constraints, and nonlinearity into the model, this dramatic run-time was expected. Due to this, generating a sufficient heuristic is worth further study.

In both models, moreover, we assumed negligible wave activity. Since these unmanned surface vehicles are deployed on the water, which has varying degrees of wave activity, introducing this aspect to the model deserves inspection. The direction, speed, and magnitude of waves would affect not only the velocity of the vessel, but also the energy required to traverse any part of the water's surface. Including velocity as a decision variable, then, would require not only understanding how much energy is mandated to achieve a given speed, but also how the energy usage changes to maintain that speed on a fluctuating water surface.

Bibliography

[1] J. E. Beasley. Route first-cluster second methods for vehicle routing, *Omega*, 11 (4): 403-408, 1983.

[2] G. Clarke and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12: 568-81, 1964.

[3] J. R. Current. Multiobjective Design of Transportation Networks. Ph.D. Thesis, Department of Geography and Environmental Engineering, The Johns Hopkins University, 1981.

[4] J. R. Current and D. A. Schilling. The Covering Salesman Problem. *Transportation Science*, 23: 208-213, 1989.

[5] J. R. Current and D. A. Schilling. The Median Tour and Maximal Covering Problems. *European Journal of Operational Research*, 73: 114-126, 1994.

[6] G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6 (1): 80-91, 1959.

[7] M.S. Daskin. *Service Science*, John Wiley and Sons, Inc., New York, 2010.

[8] K. Doerner, A. Focke and W. J. Gutjahr. Multicriteria tour planning for mobile healthcare facilities in a developing country. *European Journal of Operational Research*, 179 (3): 1078-1096, 2007.

[9] I. S. Dolinskaya and R. L. Smith. Fastest-path planning for direction-dependent speed functions. *Journal of Optimization Theory and Applications*, 158:480-497, 2013.

[10] I. S. Dolinskaya, M. Kotinis, M. G. Parsons, and R. L. Smith. Optimal short-range routing of vessels in a seaway. *Journal of Ship Research*, 53 (3): 121-129, 2009.

[11] I. S. Dolinskaya and A Maggiar. Time-optimal trajectories with bounded curvature in anisotropic media. *International Journal of Robotics Research*, 31 (14): 1761-1793, 2012.

[12] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79 (3): 497-516, 1957.

[13] M. Gendreau, G. Laporte, and F. Semet. The Covering Tour Problem. *Operations Research*, 45 (4): 568-576, 1997.

[14] B. Gillett, and L. Miller. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22: 340-349, 1974.

[15] J. Grenestedt. Personal communication.

[16] M. Hachicha, M. J. Hodgson, G. Laporte, and F. Semet. Heuristics for the multi-vehicle covering tour problem. *Computers Operations Research*, 27 (1): 29-42, 2000.

[17] M. J. Hodgson, G. Laporte and F. Semet. A covering tour model for planning mobile health care facilities in Suhum district, Ghana. *Journal of Regional Science*, 38: 621-638, 1998.

[18] N. Jozefowiez, F. Semet, and E. Talbi. The bi-objective covering tour problem. *Computers and Operations Research*, 34 (7): 1929-1942, 2007.

[19] M. Labbé and G. Laporte. Maximizing User Convenience and Postal Service Efficiency in Post Box Location. *Belgian Journal of Operations Research, Statistics and Computer Science*, 26: 21-35, 1986.

[20] G. Laporte. The vehicle routing problem: an overview of exact and approximate algorithms. *European Journal of Operational Research*, 59: 345-358, 1992.

[21] C. Malandraki and M. S. Daskin. The maximum benefit Chinese postman problem and the maximum benefit traveling salesman problem. *European Journal of Operational Research*, 65: 218-234, 1993.

[22] W. A. Oliveira, M. P. Mello, A. C. Moretti, and E. F. Reis. The multi-vehicle covering tour problem: building routes for urban patrolling. arXiv:1309.5502v1, submitted 2013.

[23] J. R. Oppong and M. J. Hodgson. Spatial Accessibility to Health Care Facilities in Suhum District, Ghana. *Professional Geographer*, 46: 199-209, 1994

[24] C. S. Revelle and G. Laporte. New Directions in Plant Location. *Studies in Locational Analysis*, 5: 31-58, 1993.

[25] R. Sahraeian and A. Ebrahimi. The allocated maximal backup covering tour problem. *Journal of Management and Sustainability*, 2 (1): 151-159, 2012.

[26] L. V. Snyder and Z. J. Shen. *Fundamentals of Supply Chain Theory*. John Wiley & Sons, Inc. Hoboken, New Jersey, 2nd edition, 2017, forthcoming.

[27] A. Wren. *Computers in Transport Planning and Operation*. Ian Allan, London, 1971.

[28] A. Wren, and A. Holliday. Computer scheduling of vehicles form one or more depots to a number of delivery points. *Operational Research Quarterly*, 23: 333-344, 1972.

Biography

Joshua T. Margolis was born in Princeton, NJ, on April 4, 1991. Along with his three brothers, Adam, Ryan, and Dylan, Joshua was raised by his mother, Cindy, and father, David, in their family home in Langhorne, PA. His uncle, Barry Lynn, was also an integral part of his upbringing.

Joshua attended George School, a Quaker coeducational boarding school in Newtown, PA, where he graduated high school in May of 2009. The following fall he attended Dickinson College in Carlisle, PA, where he graduated summa cum laude earning his Bachelor of Science in Physics and his Bachelor of Science in Mathematics in May of 2013. During his tenure there, he was inducted into Pi Mu Epsilon, Sigma Pi Sigma, and Phi Beta Kappa honor societies. His senior thesis in physics was titled "Manipulating Spatial Frequencies Through LabView."

In the fall of 2013, Joshua taught physics at Westminster School, a private boarding school in Simsbury, CT. Teaching both conceptual and AP physics, he also coached soccer and track. In the fall of 2014, Joshua came to Lehigh University and began working with Dr. Lawrence V. Snyder in order to pursue his Master of Science in Industrial and Systems Engineering.