2013

A Swarm of Salesmen: Algorithmic Approaches to Multiagent Modeling

Alexandre Amlie-Wolf
Oberlin College

A SWARM OF SALESMEN: ALGORITHMIC APPROACHES TO MULTIAGENT MODELING

ALEXANDRE AMLIE-WOLF

Abstract. This honors thesis describes the algorithmic abstraction of a problem modeling a swarm of Mars rovers, where many "agents" must together achieve a goal. The algorithmic formulation of this problem is based on the traveling salesman problem (TSP), and so in this thesis I offer a review of the mathematical technique of linear programming in the context of its application to the TSP, an overview of some variations of the TSP and algorithms for approximating and solving them, and formulations without solutions of two novel TSP variations which are useful for modeling the original problem.

Oberlin College Computer Science Honors Thesis 2013
Supervisor: Tom Wexler

1. Introduction

In this day and age of space exploration, some of the most exciting scientific projects being undertaken by humankind are those investigating Mars, the fourth planet from the sun. NASA, the National Aeronautics and Space Administration of the United States of America, has made full topographical data of Mars available to the public. These data were recorded by the Mars Orbiter Laser Altimeter aboard the Mars Global Surveyor over a period of four and a half years and 640 million elevation measurements as of June 30, 2001. [31]. These measurements have an absolute accuracy of 13 meters with respect to Mars' center of mass, meaning that the global topography of Mars is known to a greater accuracy than Earth's continents (as of 1999) [32]. To date, Mars has only been explored by single "rovers"; mobile robots that carry scientific equipment with which to conduct tests on the planet's surface, such as Curiosity, Spirit, Opportunity, and Sojourner. An alternative approach to using single rovers is to apply ideas from swarm intelligence by replacing the rover with a group ("swarm") of less individually powerful robots. Such a system would be more robust to failure and more modular, both of which are important characteristics for exploring an environment as hostile as the surface of Mars.

However, as opposed to a single robot which only has to worry about its own actions (a difficult enough problem already), a group of robots such as this would have to be made to cooperate in some sense; at the very least, they can't get in each others way.

Date: May 8, 2013.

Given that each member of the group would be less capable than the single rover by necessity, getting a group of robots such as this to accomplish anything meaningful would be a difficult problem, especially in an environment like Mars, where any control signals from Earth would take minutes to arrive. The goal of this honors project is to investigate this problem using a variety of approaches, from AI/robotics planning to algorithmic approaches (which form the bulk of the work done).

One approach taken was to use techniques from robotics and AI to try and model the swarm of rovers as a decentralized group of learning agents; that is, instead of sending out commands to each robot from some central source (i.e. Earth), the idea was to give the group as a whole some goals to accomplish, and implement some learning and possibly communication algorithm in each robot which would lead to emergent behavior that would satisfy these goals without requiring central control. This approach was highly influenced by principles of swarm intelligence, especially biologically inspired behavior such as the self-organizing behavior of insect colonies [22] or the immune system [11]. However, as another goal of the project was to develop algorithms which were amenable to precise theoretical analysis, this approach did not lend itself well to the overall goals of the project. Many of these AI planning problems are computationally intractable and often make overly simplifying assumptions which do not lend themselves well to real world applications. Additionally, most swarm intelligence algorithms cannot be theoretically guaranteed to converge on a solution, another aspect which we wanted in our design.

Thus, the other approach taken in the thesis project, which was using algorithmic and theoretical approaches to abstractions of this problem of multiagent cooperation, became the bulk of the project. Instead of considering all the complex factors such as being on Mars and the robotics control aspect of the problem, we abstracted away most of them and formulated the problem in terms of the Traveling Salesman Problem (TSP). This approach led us to investigate many variants of the TSP, and so the main thrust of this thesis is to describe some of these variants and the solution or approximation algorithms designed for them, as well as introduce some novel variants that could be used to more precisely model our problem. Although I did not end up designing algorithms and proving them for this problem, the thesis turned into more of a review on the TSP and solution/approximation approaches to it, and as this is one of the most commonly studied problems in algorithmic analysis, this turned out to be more than enough to study over the course of the honors project. I will mention that we wanted to consider polynomial time approximation algorithms, and so I leave out details of many algorithms that use optimization techniques like branch and bound to find "good enough" solutions that run in some "reasonable" amount of time.

The rest of this paper is structured as follows: In section 2, I describe our approach to investigating the problem algorithmically. This includes an overview of linear programming in section 2.1, a background of the general traveling salesman problem in section 2.2, and a categorization and description of some of the variants thereof in section 2.3.

Then, in section 3 I briefly offer descriptions of two novel TSP variants and formulations thereof. Finally, in section 4, I conclude with a discussion of the pros and cons of our theoretical abstraction and possible next steps.

2. Theoretical Abstraction

I now give a brief review of linear programming, a review of the traveling salesman problem, and an overview of some variants thereof.

2.1. **Linear Programming.**

Linear programming is a mathematical optimization technique which is used quite often in approximation and solution algorithms for various problems. I give a brief review of this topic because it is used in many algorithms that are used to approximate the TSP, which lends itself well to such a formulation.

A *linear programming (LP)* problem is the problem of maximizing or minimizing an *objective function*, which must be linear, subject to a finite number of linear constraints [14]. For example, consider the objective function $f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} c_j x_j$ for some $c_1, c_2, \ldots, c_n$, and the linear inequalities $a_1 x_1 \leq b_1$, $a_2 x_2 \leq b_2$, ..., $a_n x_n \leq b_n$, which are the constraints. Say we are trying to minimize f and we also want to make sure that every x_i must be ≥ 0. Then we can formulate this problem as the following linear program:

$$\text{minimize} \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{subject to } a_i x_i \leq b_i \ (i = 1, 2, \ldots, n) \tag{2}$$

$$x_i \geq 0 \ (i = 1, 2, \ldots, n) \tag{3}$$

This is the *standard form* of an LP problem according to Chvátal, whose style I will follow, where line 1 describes the objective function, and lines 2 and 3 describe the linear constraints. There are various methods for solving LP problems, including the *simplex method* and the *ellipsoid method*. The details of such solution methods are not important to this thesis since I will only use them as a 'black box' for solving LPs; further details may be found in [14]. Some problems may also use what is called the *integer linear program* (ILP) formulation, where all the variables in the objective function are restricted to $\mathbb{Z}$, the set of all integers. Unlike regular LPs, solving an ILP is NP-hard [14]. However, such formulations are used for many of the algorithms and variants I will discuss.

2.2. **The Traveling Salesman Problem (TSP).**

The traveling salesman problem is arguably the most well-known example of a computationally intractable problem. In its most general form, it consists of a set of cities and values representing the distances between each pair of cities, and the problem is

to find the shortest path which visits each city exactly once and return to the starting city, without reusing any edges [15]. Although the problem is technically solvable by brute force (i.e. searching all possible sequences of cities), this takes factorial time in the number of cities ($O(n!)$), meaning that for a modestly sized problem with 33 cities, there are 131,565,418,466,846,765,083,609,006,080,000,000 distinct tours. Even using a supercomputer such as the IBM Roadrunner Cluster at the United States Department of Energy, this would take about 28 trillion years, even assuming each tour can be checked with a single arithmetic operation [15]. Clearly, this is not a feasible approach.

As the TSP is one of the most intensely studied problems in combinatorial optimization, there are a huge number of solution and approximation algorithms for it and its variants. Therefore, by necessity, I limit this review to a subset of these algorithms and variants, specifically those most directly related to linear programming and used for the variants most relevant to our motivating problem of the swarm of Mars rovers. For in depth treatments on the TSP, the reader is referred to [2, 30, 25].

First, consider the formulation of the TSP. The TSP can be modeled as a fully connected graph, where the vertices represent cities and an edge has cost encoding the 'difficulty' of traveling between its endpoint cities (I say difficulty here because some formulations consider time, others consider distance, and others consider cost between cities). Then the TSP on this graph is the problem of finding a Hamiltonian cycle of shortest length [30]. In this form, there is no polynomial time approximation algorithm for the traveling salesman problem with a bounded ratio unless $P = NP$; in other words, there is no way to guarantee even an approximately good solution to the general problem.

However, research has been done on many sub-classes of the TSP, some of which yield fixed-bound polynomial time approximation algorithms. For this thesis, I am only concerned with the formulations called *metric TSP*, where edge costs satisfy the triangle inequality ($c(x, y) + c(y, z) \geq c(x, z) \forall x, y, z \in V$), and where the edges are symmetric (i.e. we have an undirected graph). I now describe a simple approach to approximating the metric TSP, which yields a 2-approximate solution, followed by a description of the classic algorithm for metric TSP, Christofides algorithm, which yields a better 1.5-approximate solution [13].

2-Approximation Algorithm. Given a undirected complete graph $G = (V, E)$ representing a metric TSP instance, compute the minimum spanning tree T of G (which is possible in linear time [18]), root T at some arbitrary vertex, and then traverse T in preorder, listing this ordering of vertices without repeats as our TSP solution A. This yields a 2-approximate solution to the metric TSP, i.e. $cost(A) \leq 2 * cost(O)$ where O is an optimal solution, and runs in polynomial time. The proof of this approximation bound is in the appendix.

Christofides' Algorithm. Given a TSP instance, a complete graph $G = (V, E)$, compute a minimum spanning tree T on G. Next, compute a perfect matching M with minimum

weight on the set of vertices with odd degree in T, and combine M and T to make a multigraph N. Finally, find an Eulerian circuit in N (a cycle in the graph which visits every edge exactly once), and convert that into an Hamiltonian cycle, corresponding to a TSP solution, by skipping visited nodes in the Eulerian circuit. This algorithm is well known to be a 3/2-approximation for the TSP [13].

Held-Karp Lower Bound. Another important result in TSP approximation that I will now describe is the family of lower bounds on the symmetric TSP first introduced by Michael Held and Richard M. Karp in 1969 [26]. This is used as part of many approximation algorithms, most relevantly the 2.5-approximation for the prize collecting TSP that I will describe later in this thesis. This section is all based on reference [26]. In the interest of space, I leave out the proofs in this derivation; they can be found in section A.2 of the appendix.

The Held-Karp lower bound uses observations about 1-trees, which are a slight variation on spanning trees, to find lower bounds on the cost of an optimum TSP tour on a graph. A 1-tree is a graph with vertices $\{1, \ldots, n\}$ and consists of a tree on the set $\{2, \ldots, n\}$ along with two distinct edges at vertex 1. 1-tree has a single cycle, which contains vertex 1, and vertex 1 always has degree two; furthermore, while it is computationally hard to solve the TSP, a minimum-weight 1-tree can be found by finding a minimum spanning tree on the vertices $\{2, \ldots, n\}$ (which can be solved in $O(n)$ time [18]), and then choosing the two lowest-weight edges at vertex 1. Then, every tour is a 1-tree, where vertex 1 is the 'depot' or 'home' city in TSP terms, and a 1-tree is a tour if and only if each of its vertices has degree 2. Therefore, if a minimum-weight 1-tree is a tour, it is the solution to the TSP. I now give the first lemma used to derive the Held-Karp lower bound, as follows:

Lemma 1. *Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ be a real number vector. If C^* is a minimum-weight tour with respect to edge weights c_{ij} on a graph, then it is also a minimum-weight tour with respect to edge weights $c_{ij} + \pi_i + \pi_j$, i.e. the original edge weights transformed by π.*

Unlike a tour, this edge weight transformation does affect the minimum spanning 1-tree. This leads us to the main idea of the Held-Karp lower bound, which is to search for a vector π to transform the edge weights by such that the minimum weight 1-tree with respect to weights $c_{ij} + \pi_i + \pi_j$ is a tour. They introduce a 'gap' function $f(\pi)$ which is defined as the difference between the cost of a minimum tour and the cost of a minimum 1-tree with respect to the edge weights $c_{ij} + \pi_i + \pi_j$ (transformed by π). The cost of the minimum tour will always equal or exceed the cost of the minimum 1-tree, since all vertices do not need to have degree 2 in a 1-tree; this is where the lower bound on the cost of the tour comes from. Therefore, $f(\pi)$ is a non-negative function. To derive the best lower bound on the cost of a TSP tour, we want to find $min_\pi f(\pi)$. This is because any transformation vector π will give us a lower bound, but we want that lower bound to be as close as possible to the actual optimal TSP tour's cost, so we want

the smallest difference between the optimal 1-tree and the tour, which is measured by $f(\pi)$. Finding $min_\pi f(\pi)$ can be expressed as a linear program, which we will see later.

The actual family of Held-Karp lower bounds comes from relating this function to a linear relaxation of this integer linear program representing the TSP: Deriving the Held-Karp lower bound comes from relating the function $f(\pi)$ to this integer linear program encoding the TSP:

$$\text{Program 1: minimize} \sum_{1\leq i<j\leq n} c_{ij}x_{ij}$$

$$\text{subject to} \sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} = 2 \text{ for } i = (1,\ldots,n) \tag{1}$$

$$\sum_{i,j\in S, i<j} x_{ij} \leq |S| - 1 \ \forall S \subset \{2,3,\ldots,n\} \tag{2}$$

$$0 \leq x_{ij} \leq 1 \ (1 \leq i < j \leq n) \tag{3}$$

$$x_{ij} \text{ integer} \tag{4}$$

The optimal solutions of this integer LP correspond to optimal TSP tours with weights c_{ij} [16, 17]; the tour corresponding to a feasible assignment of x_{ij} includes edge $\{i, j\}$ if and only if $x_{ij} = 1$. The constraints (1) ensure that each vertex in the tour has degree 2, the constraints (2) are subtour elimination constraints, ensuring that no cycle is formed in a (proper) subset of the vertices since the only cycle in a TSP must include the depot, and the other constraints are on the variables in the ILP. I will now derive the relationship between the *linear relaxation* of this ILP (where the variables are now allowed to be real numbers instead of integers) and minimizing the function $f(\pi)$.

Lemma 2. *In program 1, it is possible to replace constraints* (1) *by*
$\sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} = 2, i = 1, 2, \ldots, n-1,$ *and* $\sum_{1\leq i<j\leq n} x_{ij} = n$

This allows us to rewrite program 1 as

$$\text{Program 1': minimize} \sum_{1\leq i<j\leq n} c_{ij}x_{ij}$$

$$\text{subject to} \sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} = 2, \text{ for } i = (2,3,\ldots,n-1) \tag{1}$$

$$\sum_{j} x_{1j} = 2 \tag{2}$$

$$\sum_{1\leq i<j\leq n} x_{ij} = n \tag{3}$$

$$\sum_{i,j\in S, i<j} x_{ij} \leq |S| - 1 \forall S \subset \{2,3,\ldots,n\} \tag{4}$$

$$x_{ij} \leq 1 \tag{5}$$

$$x_{ij} \geq 0 \tag{6}$$

$$x_{ij} \text{ integer} \tag{7}$$

Note that constraints (5) and (6) are now split up; this is to make the next step of the derivation easier. To make a more compact representation of this linear program, which is still the TSP ILP, define x and c to be $\binom{n}{2}$-sized vectors made up of all x_{ij} and $c_{ij} (1 \leq i < j \leq n)$, respectively. This allows us to replace the constraints (1) with $Ax = b$ where A is a matrix defined to represent the summation in the constraints and b is a vector made up entirely of 2s, and the constraints (2), (3), (4), and (5) with $A'x \leq b'$, where A' and b' are similarly defined to represent the constraints (although these constructions are slightly more complex). This allows us to rewrite the program 1' as the more compact

$$\min_x \{cx | Ax = b, A'x \leq b', x \geq 0, x \text{ integer}\}$$

I now take advantage of the following theorem relating 1-trees to extreme points of a polyhedron. Details on its derivation are found in section 6 of [26], and involve a theorem about matroids, a type of structure, which yields this theorem as a special case.

Theorem 3. *Let $T^1, T^2, \ldots, T^k, \ldots, T^q$ be the 1-trees defined on the vertex set $\{1, 2, \ldots, n\}$. For each T^k, define an $\binom{n}{2}$-vector called (e^k_{ij}) as:*

$$(e^k_{ij}) = \begin{cases} 1 & \text{if } (i,j) \text{ is an edge of } T^k \\ 0 & \text{otherwise} \end{cases}$$

Then the extreme points of the polyhedron $A'x \leq b'$, $x \geq 0$ are the points (e^k_{ij}).

Using this theorem, the problem of finding a minimum 1-tree with respect to some $\binom{n}{2}$-sized weights vector c can be represented as a linear program of the form

$$\min_x \{cx | A'x \leq b', x \geq 0\}$$

More relevantly to this derivation, the problem of finding a minimum 1-tree with respect to the transformed weights $c_{ij} + \pi_i + \pi_j$ can be represented as

$$\min_x \{cx + \pi Ax | A'x \leq b', x \geq 0\}$$

Recalling the gap function $f(\pi)$ that measures the difference between the weight of a minimum (TSP) tour and a minimum 1-tree with respect to the weights $c_{ij} + \pi_i + \pi_j$, it is possible to represent $f(\pi)$ as follows (remember that b is a vector of all 2s):

$$\begin{aligned} f(\pi) &= \min_x \{cx | Ax = b, A'x \leq b', x \geq 0, x \text{ integer}\} + \pi b - \min_x \{cx + \pi Ax | A'x \leq b', x \geq 0\} \\ &= \min_x \{cx | Ax = b, A'x \leq b', x \geq 0, x \text{ integer}\} - \min_x \{cx + \pi(Ax - b) | A'x \leq b', x \geq 0\} \end{aligned}$$

Defining a function $w(\pi)$ which represents the weight of the minimum 1-tree with respect to the weights $c_{ij} + \pi_i + \pi_j$; using theorem 3, this can be written as

$$w(\pi) = \min_x \{cx + \pi(Ax - b) | A'x \leq b', x \geq 0\}$$

Then, minimizing $f(\pi)$ is equivalent to maximizing $w(\pi)$. The following theorem relates the maximization of $w(\pi)$ back to program 1, the integer LP for the TSP:

Theorem 4. *$max_\pi w(\pi) = v$, where*

$$v = min_x\{cx|Ax = b, A'x \leq b', x \geq 0\},$$

I.e. v is the linear relaxation of program 1.

This concludes the formal derivation of the Held-Karp lower bounds on the length of the traveling salesman. In summary, I have shown that the linear relaxation of program 1 can be used to derive a lower bound on the length of the optimal TSP tour, and furthermore that it is possible to derive a 'maximum' lower bound. I will use this fact in some of the approximation techniques for variations on the TSP, and this lower bound is widely used.

2.3. **Variants on the TSP.**

There are many variants on the classical TSP that can be used to model more specific problems and may lend themselves to different solution and approximation techniques. In this section I present a categorization of some of these variants, descriptions of each variant, and the algorithms used to approximate solutions to each variant.

Although this is a partial list of variants, these are some of the characteristics that could prove useful in categorizing variants I did not describe. There may also be other possible values for these characteristics (such as uniform penalties), but I left out descriptions of those that did not come up in the variants I describe here. I categorized each variant using 4 characteristics:

# of Tours: Whether there is a single salesman, or if multiple salesmen may be sent on tours.

Penalties: Infinite if no vertex may be missed (as in the classical TSP), or non-uniform punishment for missing any vertex if it is possible to visit a subset of vertices (i.e. each vertex has a specific cost associated with missing it).

Time Windows: Whether or not there are time windows on each vertex; that is, is there a time window in which a vertex must be visited?

Objective Function: This is the function describing how "good" any one solution is. In these variants, this is either minimizing the edge costs on the tour, or minimizing the edge costs on the tour minus the cost for missed vertices.

See table 1 for the overview of the variants I will now discuss in more detail, including their formulations and solution methods. These variants are the prize collecting traveling salesman (PCTSP) [8, 3, 5, 20], the multiple traveling salesman problem (mTSP) [6, 37, 9, 10, 12, 28, 1, 19, 23], and the time constrained traveling salesman problem (TCTSP) [4, 34].

Variant	# of Tours	Penalties	Time Windows	Objective Function	Sources
TSP	1	Infinite	None	Minimize Edge Cost	[13, 15, 2, 30, 25, 26, 16, 17]
PCTSP	1	Non-uniform	None	Minimize Edge Cost-Missed Prizes	[8, 3, 5, 20]
mTSP	m	Infinite	None	Minimize Edge Cost	[6, 37, 9, 10, 12, 28, 1, 19, 23]
TCTSP	1	Infinite	Yes	Minimize Edge Cost	[4, 34]

TABLE 1. List of Known Variants

2.3.1. *Prize Collecting Traveling Salesman Problem.* In the prize collecting traveling salesman problem (PCTSP), an instance of the problem consists of a complete undirected graph $G = (V, E)$, where each edge $\{i, j\} \in E$ has an associated cost $c_{\{i,j\}}$ and each vertex $i \in V$ has a non-negative penalty π_i [5]. The goal is to generate a tour that visits some subset of the vertices such that the sum of the cost of the edges of the tour plus the sum of the penalties of all the vertices not in the tour is minimized. In the most general version first introduced by Balas in 1989 [5], there is another constraint, which is that the rewards of the visited vertices must meet or exceed a quota. This problem generalizes the quota TSP, which is when the salesman must visit a certain number of cities but is not punished for those that are missed, as well as the penalty TSP, where there is no required quota of cities to be visited, but there are penalties for missing cities [3]. The PCTSP is one of the more well-studied TSP variants, and so I will go over only a subset of algorithms related to it, giving citations for further information.

Formulation: I will focus on a specific formulation, the one that yields the 2.5-approximation algorithm given by Bienstock et al. in 1991 [8]. This is the same as the general PCTSP formulation except without a quota requirement, and with the assumption that the edge costs satisfy the triangle inequality, i.e. $c_{\{i,j\}} \leq c_{\{i,k\}} + c_{\{k,j\}}$ for all $i, j, k \in V$ (so it's a *metric PCTSP*). Although there have been many advances since then, including a 2-approximation using primal dual techniques from linear programming [20] that is the basis of most more recent approaches to approximating the PCTSP that improve the bound even further [3], I will focus on this formulation and algorithm as an example of an approximation algorithm that is relatively simple to understand yet powerful enough to yield a theoretically guaranteed bound on the optimal solution.

Solution and Approximation Algorithms: I now present the 2.5-approximation for the PCTSP. Again in the interest of space, I leave the main approximation proof to the appendix, in section A.3.

In this formulation, define Z^H as the cost of the solution produced by the heuristic H. Then define Z^* as the optimal solution to the PCTSP, and more specifically, $Z^*(j)$

as the optimal solution to the PCTSP when vertex j must be in the tour. Then we have $Z^* = \min\{\sum_{i \in V} \pi_i, \min_{j \in V}\{Z^*(j)\}\}$; either we visit none of the vertices and pay all the costs, or we take the best tour we can find, whichever is cheaper.

To formulate this as an LP, define the variables y_i as one if vertex i is in the tour and 0 otherwise, and x_e as one if edge e is used in the tour and zero otherwise. Also define $\delta(S)$ as the set of edges crossing S, for any $S \subseteq V$. This allows us to define the LP for $Z^*(j)$, program $PC_1(j)$:

$$Z^*(j) = \text{ minimize } \sum_{e \in E} c_e x_e + \sum_{i \in V} \pi_i (1 - y_i)$$

$$\text{subject to } \sum_{e \in \delta(\{i\})} x_e = 2y_i \forall i \in V \tag{1}$$

$$\sum_{e \in \delta(S)} x_e \geq 2y_i \forall i \in V, S \subset V \text{ such that } |S \cap \{i, j\}| = 1 \tag{2}$$

$$0 \leq x_e \leq 1 \text{ and integer} \tag{3}$$

$$0 \leq y_i \leq 1 \text{ and integer } \forall i \neq j \tag{4}$$

$$y_j = 1 \tag{5}$$

The basic idea of this approximation algorithm is to solve the LP relaxation of program $PC_1(j)$ for each vertex j (using the ellipsoid method, which I mentioned in section 2.1), transform each of those solutions into a feasible solution to the PCTSP, constructing $|V|$ feasible solutions, and then choose the best of those, or the solution in which no vertex is visited.

Before I derive the algorithm, I must make some preliminary definitions, theorems, and lemmas that will be used in the main algorithm:

Define a program PC_2 to get the Held-Karp lower bound on the TSP (as described earlier), which is formulated as:

$$Z_{HK} = \text{ minimize } \sum_{e \in E} c_e x_e$$

$$\text{subject to } \sum_{e \in \delta(S)} x_e \geq 2 \ \forall S \subset V, S \neq \emptyset \tag{1}$$

$$\sum_{e \in \delta(\{i\})} x_e = 2 \ \forall i \in V \tag{2}$$

$$0 \leq x_e \tag{3}$$

(Note that $Z_{HK} \neq Z^{HK}$, i.e. this isn't describing a heuristic solution, just the Held-Karp lower bound).

The following theorem comes from [36, 29]:

Theorem 5.

$$Z_{HK}/L(V) \geq Z_{HK}/L^C(V) \geq 2/3$$

Where $L(V)$ is the length of the optimal tour through the vertices V and $L^C(V)$ is the length of the tour found by Christofides' algorithm through the vertices V.

Finally, we have a lemma about PC_2 that follows from a lemma from [21]:

Lemma 6. *In problem PC_2, one can ignore constraint 2 without changing the value of the lower bound.*

I refer the reader to [8, 21] for details of this proof.

Now I am ready to begin the description of the approximation algorithm itself:

If we use the ellipsoid method, we can solve the linear relaxation of program $PC_1(j)$ for each $j \in V$. Let $\bar{x}$ and $\bar{y}$ be an optimal solution to the LP relaxation of $PC_1(j)$ for some j found in this way. Define new vectors $\hat{x}$ and $\hat{y}$ according to equation 1:

$$\hat{x_e} = (5/3)\bar{x}_e \ \forall e \in E$$

$$\hat{y_i} = \begin{cases} 1 & \text{if } \bar{y}_i \geq (3/5) \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in V$$

From the definition of $\hat{y}_i$, derive equation 2: $1 - \hat{y}_i \leq (5/2)(1 - \bar{y}_i)\ \forall i \in V$. Now define $T \subseteq V$ as $T = \{i \in V | \hat{y}_i = 1\}$, and construct a tour through those vertices using Christofides' algorithm. This is the tour produced by our algorithm, when j must be in the tour. We call this the modified linear program (MLP) heuristic on vertex j, denoted by $Z^{MLP}(j)$. Then, this solution pays penalty costs for all vertices not in T, so we have

$$Z^{MLP}(j) = L^C(T) + \sum_{i \in V} \pi_i(1 - \hat{y}_i)$$

for the tour where $j \in V$ is in the tour. Then, the MLP heuristic, denoted as Z^{MLP}, chooses the best solution over all vertices or the one where no vertices are taken; that is,

$$Z^{MLP} = \min\{\sum_{i \in V} \pi_i, \min_{j \in V}\{Z^{MLP}(j)\}\}$$

Theorem 7. $Z^{MLP}/Z^* \leq 2.5$.

Therefore, I have described a 2.5-approximation algorithm for the PCTSP which relaxes the integer constraints on an integer LP and then "rounds" those fractional solutions to an integer solution and argues that those integer solutions are close to the optimal solution. This is useful as a good example of an approach that is useful for approximately solving problems that are otherwise intractable.

2.3.2. *Multiple Traveling Salesman Problem.* The multiple traveling salesman problem (mTSP) is a less well-studied variant of the TSP [6], where the main difference is that there can be more than one salesman/tour in the solution. I now briefly review a formulation for this problem and approaches to solving or approximating this problem.

Formulation: The mTSP can be defined similarly to the TSP: given a graph $G = (V, E)$ where the vertices represent cities and the edges have costs representing the 'difficulty' of traveling between cities, the goal is to make a tour of the cities and minimize the cost of the edges on the tour. The difference in the mTSP is that there are now m salesman which can all make tours, all starting from some 'home' or 'depot' node, and so the goal is to find m tours, one for each salesman, where each non-depot node still is only visited once and the total cost of edges on every tour is minimized. This problem has real world applications that the TSP cannot necessarily be directly used for, such as mission planning for autonomous mobile robots [37, 9, 10]. It is also worth noting that the TSP is a special case of the mTSP where $m = 1$, so any solution techniques for mTSP can be used for TSP.

There are many integer linear programming formulations for the mTSP, including assignment-based formulations, which seem to be the norm [6], but also including a k-degree center tree-based formulation, and a flow-based formulation, both from [12] and based on the vehicle routing problem. I will focus on the assignment-based formulation, as it is the most closely related to the types of problems and solution methods I describe in this thesis.

To describe the assignment-based formulation, define the following variable:

$$x_{i,j} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is taken on any tour} \\ 0 & \text{otherwise} \end{cases}$$

Then one general formulation is as follows:

$$\text{minimize} \sum_{1 \le i < j \le n} c_{ij} x_{ij}$$

$$\text{subject to} \sum_{j=2}^{n} x_{1j} = m \tag{1}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{2}$$

$$\sum_{1 \le i < j \le n} x_{ij} = 2 \tag{3}$$

$$\text{subtour elimination constraints} \tag{4}$$

$$x_{ij} \in \{0, 1\} \forall \{i, j\} \in E \tag{5}$$

The only difference from the normal ILP for the TSP is that the first two constraints now stipulate that m tours must leave from and end at the 'depot' node. The "subtour elimination constraints" are similar to those in the other LP formulations we have seen; these constraints make sure that no cycles exist in the graph except for the main one(s) which include the 'depot' node. Many different ones have been proposed in the mTSP literature, but these are outside the scope of this thesis so I leave them out.

Solution and Approximation Approaches: There have been three main approaches to solving the mTSP: exact algorithms, heuristic approaches, and transformations to the TSP. Since these are not closely related to the goal of this thesis (finding provable approximations), I leave out most details of these approaches and refer to the review paper [6] for further details. Although this variant did not offer much in the way of useful algorithms given the goal of this thesis, it is a good example of a variant for which other approaches have been tested and provable approximations have not been studied or found. I now present a brief overview of the various solution techniques with some citations:

Exact Solutions: Integer linear programming formulations, cutting plane techniques (relaxing some of the constraints on the problem similarly to the PCTSP algorithm described earlier) [28] and branch and bound optimization techniques (where lower bounds on the solution are obtained, similarly to the Held-Karp lower bound described earlier) [1, 19, 23]. These are concerned with optimizing the 'brute-force' search for exact solutions, but do not necessarily come within a provable bound of the optimal solution.

Heuristics: Evolutionary algorithms, simulated annealing [33], Tabu search, genetic algorithms [38], and neural networks [35]. These use heuristic approaches to solve the problem but cannot be theoretically guaranteed to converge or find any bounded near-optimal solution.

Transformations to TSP: Asymmetric mTSP to asymmetric TSP [7], symmetric mTSP to symmetric TSP [27], and multidepot mTSP to TSP [24]. These transform the mTSP to the regular TSP, which may provide inspiration for our multi-agent problem, but these algorithms only deal with the transformation to TSP, not necessarily solving it thereafter.

2.3.3. *Time Constrained Traveling Salesman Problem.* This section describes the time constrained traveling salesman problem (TCTSP). This is another example of a variant for which only exact algorithms are known (with 'good enough' running times obtained by optimization techniques) other than for special cases which are irrelevant to this thesis, and so I will again not go into too much detail on these solutions.

Similarly to the TSP, an instance of the TCTSP consists of a graph where the vertices represent cities and the edges between vertices have costs representing the time required to travel between each city. The goal is to compute a (single salesman) tour for that visits every city exactly once such that the cost of the tour is minimized, with the additional constraint that the visit to each city must be made within specified time windows. If t_i is when the salesman visits city i, then there is a constraint such that $l_i \leq t_i \leq u_i$,

where l_i and u_i are the lower and upper time bounds for that city, respectively [4]. Note that this formulation describes 'hard' time bounds, i.e. it is not possible to miss the time windows. Some other variants may use 'soft' time windows, where the salesman can pay some fee to visit a city outside of the time window.

Formulation: I describe the formulation from Edward Baker, 1982 [4]. Instead of the typical binary variable x_{ij}, define a single variable t_i to be the time that city i is visited; this necessitates an additional variable t_{n+1} to describe when the home city is visited again at the end of the tour. Then, D is the nonnegative time matrix, where d_{ij} is the time to travel from city i to city j. Again this is the metric TSP, where the time costs satisfy the triangle inequality. Then the problem can be described as the following LP:

$$\text{Minimize } t_{n+1} - t_1$$

$$t_i - t_1 \geq d_{1i} \; i = 2, 3, \ldots, n \tag{1}$$

$$|t_i - t_j| \geq d_{ij} \; i = 3, 4, \ldots, n, 2 \leq j < i \tag{2}$$

$$t_{n+1} - t_i \geq d_{i1} \; i = 2, 3, \ldots, n \tag{3}$$

$$t_i \geq 0 \forall i \tag{4}$$

$$l_i \leq t_i \leq u_i \; i = 2, 3, \ldots, n \tag{5}$$

Clearly, constraints 1-4 define a traveling salesman problem. See [4] for a more formal proof, but constraints 1 and 3 guarantee that the smallest and largest t_i values correspond to the departure from and return to the depot city, respectively, and constraints 2 guarantee that each city is assigned a unique visitation time and the difference between any two cities visitation times is at least the distance between those cities. Then, constraints 5 are the only ones unique to the TCTSP formulation.

Solution and Approximation Approaches: Since introducing time constraints to the TSP only makes it harder in its most general form, many approaches have been to try and solve the problem 'practically'; that is, using techniques like branch and bound algorithms to solve it optimally in some 'reasonable' amount of time. It is sometimes possible to solve using theoretically guaranteed polynomial time algorithms when it is restricted to special cases, such as when the cities are placed along a straight line [34]. However, since this thesis is concerned with the specific multi-agent learning problem described earlier, where nothing may be assumed about where the 'cities' (or whatever the rovers need to visit) are, these special cases are irrelevant to the thesis. Unfortunately, then, this is another example of a variant which does not have theoretically guaranteed approximation algorithms, and so does not inform the type of analysis this thesis deals with.

3. Novel TSP Variants

In this section, I describe some ideas for novel TSP formulations that may be used to model our original swarm of Mars rover problem. These variations are interesting for

thinking about modeling the original problem, and the obvious next step in this line of research would be to delve more deeply into solution algorithms for them.

I now present brief descriptions of each variant, along with potential formulations.

3.0.4. *Quota Multiple Prize Collecting Traveling Salesman Problem.* The quota multiple prize collecting traveling salesman problem (qmPCTSP) generalizes the PCTSP by allowing for multiple tours, and also adds a quota, as in the quota PCTSP formulation. This quota reflects a minimum amount of prize that must be collected by the m tours.

Formulation: Since there now are multiple traveling salesmen, define new variables:

$$x_{e,k} = \begin{cases} 1 & \text{if agent } k \text{ takes edge } e \\ 0 & \text{otherwise} \end{cases}$$

and

$$y_{i,k} = \begin{cases} 1 & \text{if agent } k \text{ visits vertex } i \\ 0 & \text{otherwise} \end{cases}$$

Call M our set of m agents and q the quota. Then an LP formulation for this problem is as follows:

$$\text{minimize} \sum_{k \in M} \{ \sum_{e \in E} c_e x_{e,k} + \sum_{i \in V} \pi_i (1 - y_{i,k}) \}$$

$$\text{subject to} \sum_{e \in \delta(\{i\})} x_{e,k} = 2y_{i,k} \ \forall i \in V, k \in M \tag{1}$$

$$\sum_{e \in \delta(S)} x_{e,k} \geq 2y_{i,k} \ \forall i \in V, k \in M, S \subset V \text{ such that } |S \cap \{i, j\}| = 1 \tag{2}$$

$$\sum_{k \in M} \sum_{i \in V} \pi_i y_{i,k} \geq q \tag{3}$$

$$0 \leq \sum_{k \in M} x_{e,k} \leq 1 \tag{4}$$

$$0 \leq \sum_{k \in M} y_{i,k} \leq 1 \tag{5}$$

Constraints 1 and 2 are the same as in the normal PCTSP formulation except they apply to all m agents, constraint 3 defines the quota requirement, and constraints 4 and 5 guarantee that each vertex and edge is visited by at most one tour.

3.0.5. *Budget Multiple Prize Collecting Traveling Salesman Problem.* The budget multiple prize collecting traveling salesman problem (bmPCTSP) is similar to the qmPCTSP except that instead of a quota on the amount of prize that must be collected, there is a budget for each agent (or a total budget) describing the maximum allowable cost of the tour.

Formulation: This formulation is similar to the qmPCTSP formulation except that instead of a quota q, I define a budget b_k for each agent k representing what the most that agent can "pay" to take its tour is. To describe a total budget, we could define b, where $\sum_{k \in M} b_k = b$, but for the motivating problem of this thesis we would probably want to define individual budgets for each agent.

$$\text{minimize} \sum_{k \in M} \{ \sum_{e \in E} c_e x_{e,k} + \sum_{i \in V} \pi_i (1 - y_{i,k}) \}$$

$$\text{subject to} \sum_{e \in \delta(\{i\})} x_{e,k} = 2y_{i,k} \ \forall i \in V, k \in M \tag{1}$$

$$\sum_{e \in \delta(S)} x_{e,k} \geq 2y_{i,k} \ \forall i \in V, k \in M, S \subset V \text{ such that } |S \cap \{i, j\}| = 1 \tag{2}$$

$$\sum_{e \in E} c_e x_{e,k} \leq b_k \ \forall k \in M \tag{3}$$

$$0 \leq \sum_{k \in M} x_{e,k} \leq 1 \tag{4}$$

$$0 \leq \sum_{k \in M} y_{i,k} \leq 1 \tag{5}$$

Constraints 1 and 2 are the same as the normal PCTSP formulation, constraint 3 defines the budget requirement, and constraints 4 and 5 guarantee that each vertex and edge is visited by at most one tour.

4. Conclusion

In this thesis, I have presented the traveling salesman problem and variants thereof as a feasible method for modeling and investigating the original problem I described, the problem of getting a "swarm" of Mars rovers to cooperate in some sense. Although I did not reach the stage of designing algorithms to solve the algorithmic abstractions I defined, let alone converting those algorithms into an actual plan for the "swarm" that could be tested in simulation due to time constraints, I believe that pursuing this algorithmic approach would be a feasible and interesting way to investigate such a problem. The novel variants I described could lead to interesting results both in simulating a "swarm" as described and from the algorithmic standpoint of designing algorithms to solve such problems, and such an approach could easily be translated into the original problem. For example, the "budget" in the bmPCTSP could represent the amount of power each rover in the "swarm" is given, and the prizes on each vertex could represent the scientific utility of various locations in the environment being explored; then, a solution to the bmPCTSP could be used to make a plan for where each robot would travel in their explorations. Although I did not get to develop any novel solutions,

this honors thesis was worthwhile in getting me to that point, and clearly the next step would be to design some algorithms to solve the novel variants I presented and eventually apply those algorithms to the real world problem.

References

[1] A Iqbal Ali and Jeff L Kennington. The asymmetric m-travelling salesmen problem: A duality based branch-and-bound algorithm. *Discrete Applied Mathematics*, 13(23):259 – 276, 1986.

[2] D.L. Applegate. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press, 2006.

[3] Giorgio Ausiello, Vincenzo Bonifaci, Stefano Leonardi, and Alberto Marchetti-Spaccamela. Prize-collecting traveling salesman and related problems. *TF Gonzalez (eds), Handbook of Approximation Algorithms and Metaheuristics, CRC Press (2007) pp. 40.1*, 40, 2007.

[4] Edward K. Baker. An exact algorithm for the time-constrained traveling salesman problem. *Operations Research*, 31(5):pp. 938–945, 1983.

[5] Egon Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[6] Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209 – 219, 2006.

[7] Mandell Bellmore and Saman Hong. Transformation of multisalesman problem to the standard traveling salesman problem. *J. ACM*, 21(3):500–504, July 1974.

[8] Daniel Bienstock, Michel X. Goemans, David Simchi-Levi, and David Williamson. A note on the prize collecting traveling salesman problem. *Math. Program.*, 59(3):413–420, May 1993.

[9] B.L. Brumitt and A. Stentz. Grammps: a generalized mission planner for multiple mobile robots in unstructured environments. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1564–1571 vol.2, May.

[10] Barry Brummit and Anthony (Tony) Stentz. Dynamic mission planning for multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 1996.

[11] Arvind K. Chavali, Erwin P. Gianchandani, Kenneth S. Tung, Michael B. Lawrence, Shayn M. Peirce, and Jason A. Papin. Characterizing emergent properties of immunological systems with multi-cellular rule-based computational modeling. *Trends in Immunology*, 29(12):589 – 599, 2008.

[12] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981.

[13] Nikos Christofides. *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem.* Management sciences research report. Defense Technical Information Center, 1976.

[14] V. Chvátal. *Linear Programming*. Series of Books in the Mathematical Sciences. W. H. Freeman, 1983.

[15] W.J. Cook. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation.* Princeton University Press. Princeton University Press, 2011.

[16] G Dantzig, R Fulkerson, and S Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

[17] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. On a linear-programming, combinatorial approach to the traveling-salesman problem. *Operations Research*, 7(1):pp. 58–66, 1959.

[18] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[19] Bezalel Gavish and Kizhanathan Srikanth. An optimal solution method for large-scale multiple traveling salesmen problems. *Operations Research*, 34(5):pp. 698–717, 1986.

[20] Michel Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM JOURNAL ON COMPUTING*, 24:296–317, 1992.

[21] Michel X. Goemans and Dimitris J. Bertsimas. On the parsimonious property of connectivity problems. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, SODA '90, pages 388–396, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[22] Deborah M. Gordon. The organization of work in social insect colonies. *Complex.*, 8(1):43–46, September 2002.

[23] J. Gromicho, J. Paixo, and I. Bronco. Exact solution of multiple traveling salesman problems. In Mustafa Akgl, HorstW. Hamacher, and Sleyman Tfeki, editors, *Combinatorial Optimization*, volume 82 of *NATO ASI Series*, pages 291–292. Springer Berlin Heidelberg, 1992.

[24] Yang GuoXing. Transformation of multidepot multisalesmen problem to the standard travelling salesman problem. *European Journal of Operational Research*, 81(3):557 – 560, 1995.

[25] G. Gutin and A.P. Punnen. *The Traveling Salesman Problem and Its Variations*. Combinatorial Optimization. Springer, 2002.

[26] Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):pp. 1138–1162, 1970.

[27] Saman Hong and Manfred W. Padberg. A note on the symmetric multiple traveling salesman problem with fixed charges. *Operations Research*, 25(5):pp. 871–874, 1977.

[28] Gilbert Laporte and Yves Nobert. A cutting planes algorithm for the m-salesmen problem. *The Journal of the Operational Research Society*, 31(11):pp. 1017–1023, 1980.

[29] David B. Shmoys and David P. Williamson. Analyzing the held-karp tsp bound: a monotonicity property with application. *Information Processing Letters*, 35(6):281 – 285, 1990.

[30] D.B. Shmoys, J.K. Lenstra, A.H.G.R. Kan, and E.L. Lawler. *The Traveling Salesman Problem*. Wiley Interscience Series in Discrete Mathematics. John Wiley & Sons, 1985.

[31] D. Smith, G. Neumann, R. E. Arvidson, E. A. Guinness, and S. Slavney. Mars global surveyor laser altimeter mission experiment gridded data record. *NASA Planetary Data System*, MGS-M-MOLA-5-MEGDR-L3-V1.0, 2003.

[32] David E. Smith, Maria T. Zuber, Sean C. Solomon, Roger J. Phillips, James W. Head, James B. Garvin, W. Bruce Banerdt, Duane O. Muhleman, Gordon H. Pettengill, Gregory A. Neumann, Frank G. Lemoine, James B. Abshire, Oded Aharonson, C. David, Brown, Steven A. Hauck, Anton B. Ivanov, Patrick J. McGovern, H. Jay Zwally, and Thomas C. Duxbury. The global topography of mars and implications for surface evolution. *Science*, 284(5419):1495–1503, 1999.

[33] Chi-Hwa Song, Kyunghee Lee, and Won Don Lee. Extended simulated annealing for augmented tsp and multi-salesmen tsp. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 2340–2343 vol.3, July.

[34] John N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22:263–282, 1992.

[35] E. Wacholder, J. Han, and R.C. Mann. A neural network algorithm for the multiple traveling salesmen problem. *Biological Cybernetics*, 61:11–19, 1989.

[36] LaurenceA. Wolsey. Heuristic analysis, linear programming and branch and bound. In V.J. Rayward-Smith, editor, *Combinatorial Optimization II*, volume 13 of *Mathematical Programming Studies*, pages 121–134. Springer Berlin Heidelberg, 1980.

[37] Zhong Yu, Liang Jinhai, Gu Guochang, Zhang Rubo, and Yang Haiyan. An implementation of evolutionary computation for path planning of cooperative mobile robots. In *Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on*, volume 3, pages 1798–1802 vol.3.

[38] Tiehua Zhang, W.A. Gruver, and M.H. Smith. Team scheduling by genetic search. In *Intelligent Processing and Manufacturing of Materials, 1999. IPMM '99. Proceedings of the Second International Conference on*, volume 2, pages 839–844 vol.2.

Appendix A. Theorem Proofs from Derivations

A.1. **Proof of the 2-Approximate TSP Algorithm.** Note: this proof is based primarily on the proof from the approximation algorithms handout from CSCI 280 in Spring 2012, taught by Alexa Sharp. I tried to change some of the wording so that it wasn't exactly the same, but I realize it is very similar. There is no plagiarism intended, I just wanted to include this proof for completeness.

Lemma 8. *The algorithm produces an ordering of the vertices corresponding to a feasible TSP tour.*

Proof. Since we traverse the minimum spanning tree in preorder, clearly we traverse every vertex; then, we remove every repeated vertex from the ordering and so we have listed every vertex exactly once in our ordering. □

Lemma 9. *The solution produced by the algorithm is 2-approximate; i.e. if A is our solution and O is an optimal solution, then $cost(A) \leq 2cost(O)$.*

Proof. First we claim that the cost of the optimal ordering $O = v_{o_1}, v_{o_2}, \ldots, v_{o_n}$ is greater than or equal to the cost of T, our minimum spanning tree. We can see this by making another spanning tree T' by removing the last edge from O; since all edges have positive cost, $cost(T') < cost(O)$, and since T is a minimum spanning tree, we have $cost(T) \leq cost(T')$ by definition, so $cost(T) < cost(O)$.

Now we claim that the cost of our solution A is at most twice the cost of T. If we define W to be the full walk around T in preorder, clearly $cost(W) = 2 * cost(T)$ since W traverses every edge twice by definition of a full walk. Our solution ordering A is a subsequence of W, since it is W without the repeated vertices. Because we have a metric TSP formulation, whenever A skips repeats, it is taking a 'shortcut'. That is, if A skips over the repeats between two vertices x and z, it pays $c(x, z)$ instead of the sum of the cost of all the edges between x and z, and since the triangle inequality says $c(x, x') + c(x', z) \geq c(x, z)$ for all the neighbors x' that A would have to go over, taking this 'shortcut' is at least as cheap as taking all the edges. Therefore, we have $cost(A) \leq cost(W) = 2 * cost(T)$.

Combining these two claims, we have $cost(A) \leq 2 * cost(T) \leq 2 * cost(O)$, so A is a 2-approximation for metric TSP. □

A.2. **Held Karp with Proofs.**

Lemma 10. *Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ be a real number vector. If C^* is a minimum-weight tour with respect to edge weights c_{ij} on a graph, then it is also a minimum-weight tour with respect to edge weights $c_{ij} + \pi_i + \pi_j$, i.e. the original edge weights transformed by π.*

Proof. If we have any tour C, each vertex in the graph is included in two edges of C, by the definition of a tour. A tour with respect to the original weights c_{ij} has

weight $\sum_{\{i,j\}\in C} c_{ij}$ and a tour with respect to the modified weights $c_{ij} + \pi_i + \pi_j$ has weight $\sum_{\{i,j\}}(c_{ij} + \pi_i + \pi_j)$. Since each vertex i is included in two edges of C, the difference between these tour weights is $2\sum_i^n \pi_i$, and so changing the weights on the graph affects any generic tour C by the same amount and so a minimum-weight tour with respect to edge weights c_{ij} is also a minimum-weight tour with respect to edge weights $c_{ij} + \pi_i + \pi_j$. □

Lemma 11. *In program 1, it is possible to replace constraints* (1) *by*

$$\sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} = 2, i = 1, 2, \ldots, n-1, \text{ and } \sum_{1\leq i<j\leq n} x_{ij} = n$$

Proof.

$$\begin{aligned}
\sum_{1\leq i<j\leq n} x_{ij} &= (1/2)\sum_{i=1}^{n}(\sum_{j>i} x_{ij} + \sum_{j<i} x_{ij}) \text{ by definition} \\
&= (1/2)\sum_{i=1}^{n-1}(\sum_{j>i} x_{ij} + \sum_{j<i} x_{ij}) + (1/2)\sum_{j} x_{jn} \text{ by sum expansion} \\
&= (1/2)\sum_{i=1}^{n-1} 2 + (1/2)\sum_{j} x_{jn} \text{ by using the first property replacement in the lemma} \\
&= n - 1 + (1/2)\sum_{j} x_{jn} \text{ by sum properties}
\end{aligned}$$

So given the other constraints, it is clear that $\sum_{i\leq i<j\leq n} x_{ij} = n$ if and only if $\sum_j x_{jn} = 2$ (as can be seen in the second to last step of the derivation above). This is sort of a strange proof since it uses part of it in its own proof, but the point is that one part of the lemma implies the other and so if we can use $\sum_{1\leq i<j\leq n} x_{ij} = n$ then we can use $\sum_{j>i} x_{ij} + \sum_{j<i} x_{ji} = 2, i = 1, 2, \ldots, n-1$. □

Theorem 12. *Let $T^1, T^2, \ldots, T^k, \ldots, T^q$ be the 1-trees defined on the vertex set $\{1, 2, \ldots, n\}$. For each T^k, define an $\binom{n}{2}$-vector called (e_{ij}^k) as:*

$$(e_{ij}^k) = \begin{cases} 1 & \text{if } (i,j) \text{ is an edge of } T^k \\ 0 & \text{otherwise} \end{cases}$$

Then the extreme points of the polyhedron $A'x \leq b'$, $x \geq 0$ are the points (e_{ij}^k).

Proof. Here I will only prove that the points (e_{ij}^k) are in the polyhedron $A'x \leq b'$, $x \geq 0$, i.e. that if we assume that the theorem holds (see section 6 of [26] for the derivation of this theorem, which is a special case of another matroid theorem), I will show how it applies to our derivation. I list the constraints represented in the program represented by the polyhedron $A'x \leq b', x \geq 0$ and how the points (e_{ij}^k) satisfy them:

2. Vertex 1 always has degree 2 in a 1-tree.

3. Every 1-tree has n edges by definition.
4. The only cycle in a 1-tree contains vertex 1.
5. Trivially.

□

Theorem 13. *$max_\pi w(\pi) = v$, where*

$$v = min_x\{cx|Ax = b, A'x \leq b', x \geq 0\},$$

I.e. v is the linear relaxation of program 1.

Proof. We have defined v as the linear relaxation of program 1, and we want to show that it is equivalent to $\max_\pi w(\pi)$. If we dualize v, we obtain

$v = \max_{u,u'}\{-ub - u'b'|uA + u'A' \geq -c, u' \geq 0\}$

$v = \max_u[max_{u'}\{-ub - u'b'|uA + u'A' \geq -c, u' \geq 0\}]$ by maximization properties

If we then dualize the inner maximization problem considering u to be held constant, we get

$$v = \max_u[\min_x\{cx + u(Ax - b)|A'x \leq b', x \geq 0\}]$$

We have already seen that the inner part of this equation, with $u = \pi$ is the same as $w(\pi)$, and so we have shown that $v = \max_\pi w(\pi)$. □

A.3. **2.5-Approximation Bound for PCTSP.**

Theorem 14. $Z^{MLP}/Z^* \leq 2.5$.

Proof. We can just show that $Z^{MLP}(j)/Z^*(j) \leq 2.5$ for every $j \in V$. To do this, we define a program PC_3 which yields the Held-Karp lower bound on the subset of vertices T:

$$\text{minimize} \sum_{e\in E} c_e x_e$$

$$\text{subject to} \sum_{e\in\delta(\{S\})} x_e \geq 2 \ \forall S \subset V \text{ s.t. } T \cap S \neq \emptyset, T \cap (V \setminus S) \neq \emptyset \tag{1}$$

$$\sum_{e\in\delta(\{i\})} x_e = 2 \ \forall i \in T \tag{2}$$

$$\sum_{e\in\delta(\{i\})} x_e = 0 \ \forall i \notin T \tag{3}$$

$$x_e \geq 0 \ \forall e \in E \tag{4}$$

By applying lemma 6, we see that the solution to program PC_3 is unchanged if we remove constraints 2 and 3. Let problem PC_4 be PC_3 without those constraints, and denote its optimal solution as $\acute{x}$. Then by Theorem 5, we have that

$$L^C(T) \leq 3/2 \sum_{e\in E} c_e \acute{x}_e$$

(equation 3). We now show that our solution $\hat{x}$ is feasible as a solution for PC_4. Clearly it satisfies constraint 4, and so we just need to show it satisfies constraints 1. Consider any $S \subset V$ such that $i \in T \cap S$ and $j \in T \setminus S$. By the feasibility of $\hat{x}$ in program PC_1 and the definition of T, using constraint 2 from PC_1 and equation 1, we have that

$$\sum_{e \in \delta(S)} \hat{x}_e \geq 2\hat{y}_i \geq 2(3/5) = (6/5) \; \forall S \subset V \text{ such that } T \cap S \neq \emptyset, T \cap (V \setminus S) \neq \emptyset$$

Therefore, for any $S \subset V$ satisfying those constraints, we have

$$\sum_{e \in \delta(S)} \hat{x_e} = (5/3) \sum_{e \in \delta(S)} \bar{x_e} \geq 2$$

and so $\hat{x}$ satisfies constraint 1 from PC_4. Also, since $\acute{x}$ is optimal, we have equation 4: $\sum_{e \in E} c_e \hat{x_e} \geq \sum_{e \in E} c_e \acute{x_e}$. Now we can prove our statement:

$$\begin{aligned}
Z^{MLP}(j) &= L^C(T) + \sum_{i \in V} \pi_i (1 - \hat{y_1}) \\
&\leq 3/2 \sum_{e \in E} c_e \acute{x_e} + \sum_{i \in V} \pi_i (1 - \hat{y_i}) \text{ from equation 3} \\
&\leq 3/2 \sum_{e \in E} c_e \hat{x_e} + \sum_{i \in V} \pi_i (1 - \hat{y_i}) \text{ from equation 4} \\
&\leq 3/2 \sum_{e \in E} c_e (5/3) \bar{x_e} + (5/2) \sum_{i \in V} \pi_i (1 - \bar{y_i}) \text{ from equations 1, 2} \\
&= (5/2) \{ \sum_{e \in E} c_e \bar{x_e} + \sum_{i \in V} \pi_i (1 - \bar{y_i}) \} \text{ by algebra} \\
&\leq (5/2) Z^*(j)
\end{aligned}$$

□