ALGORITHMS FOR AN UNMANNED VEHICLE PATH PLANNING PROBLEM

A Thesis

by

JIANGLEI QIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2013

Major Subject: Mechanical Engineering

ABSTRACT

Unmanned Vehicles (UVs) have been significantly utilized in military and civil applications over the last decade. Path-planning of UVs plays an important role in effectively using the available resources such as the UVs and sensors as efficiently as possible. The main purpose of this thesis is to address two path planning problems involving a single UV.

The two problems we consider are the quota problem and the budget problem. In the quota problem, the vehicle has to visit a sufficient number of targets to satisfy the quota requirement on the total prize collected in the tour. In the budget problem, the vehicle has to comply with a constraint of the distance traveled by the UV. We solve both these problems using a practical heuristic called the prize-multiplier approach. This approach first uses a primal-dual algorithm to first assign the targets to the UV. The Lin – Kernighan Heuristic (LKH) is then applied to generate a tour of the assigned targets for the UV. We tested this approach on two different vehicle models. One model is a simple vehicle which can move in any direction without a constraint on its turning radius. The other model is a Reeds-Shepp vehicle. We also modeled both problems in C++ using the multi-commodity flow formulations, and solved them to optimality by using the Concert Technology of CPLEX.

We used the results generated by CPLEX to determine the quality of the solutions produced by the heuristics. By comparing the objective values of the obtained solutions and the running times of the heuristics and CPLEX, one can conclude that the proposed heuristics produce solutions with good quality to our problems within our desired time limits.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Rathinam for his patient and constant guidance, support and motivation throughout this research. I would also like to express my gratitude to Dr. Swaroop and Dr. Wang for being an important part of my committee.

I wish to thank my friends, Jung and Kaarthik for helping when I met troubles and doubts in the research. Thanks also go to the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my mother and father for their encouragement and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation

Unmanned Vehicles (UVs) have been utilized in civil and military applications for decades. For example, there are a series of military robots named as Packbots deployed in military operations in Afghanistan in 2002. These robots have been used to detect and disarm explosive devices. It has also been used to collect air samples to detect chemical and radiological agents[10].

Like Packbots, many other UVs are deployed to complete tasks which could be dangerous for human operators. The main advantage of deploying UVs is that they help to save on labor cost, reduce risk to human lives, and extend the operation area to places that are not suitable for humans.

Path-planning plays an important role in UV monitoring missions. The research on planning a suitable traveling path for a UV to fulfill its task while satisfying a number of resource constraints has attracted numerous research groups from around the world. In this thesis, we consider two path planning problems for a single UV tasked to visit a collection of targets. In the next section, we motivate these path planning problems in the context of surveillance applications.

1.2 Motivation and Problem Statements

Consider a research lab focused on studying the effects of climate change upon the environment by analyzing the integrity of soil, water, and air in a certain area. A number of monitoring spots (utilized as information resources) have been set up all over this area. A UV will be sent out from the lab for monitoring this area. The UV will collect the data available at each monitoring spot it visits, and return to the lab with the collected data.

Suppose the lab can generate a traveling path for the UV before sending it out by choosing the spots to be visited and planning the sequence of spots along the traveling path. Each monitoring spot is assigned a value by evaluating the importance of research data it can provide. To generate an optimal path for the UV, these values along with the coordinates of all the monitoring spots are input to the central computer in the lab through human-computer interface. The coordinates of the depot and terminal, which is the lab, are also input to the central computer.

In this scenario, consider the following path planning problems.

(1) Quota Problem

The lab does not need data from every monitoring spots in the area. However, in order to maintain decent research quality, the lab only has to sample sufficient number of monitor spots, whose total value meets the given quota. The condition of the environment revealed by processing and analyzing the data collected from the chosen monitoring spots will be adequate to represent the general environmental conditions in this area. The objective of path-planning problem in this scenario is to minimize the total traveling time of UV while visiting enough monitor spots to satisfy the quota requirement.

(2) Budget Problem

Instead of the quota, the fuel efficiency is the main concern in budget problem. Obviously, the fuel tank of UV has a size limitation. One full tank of fuel carried by UV can only support it to cover a limited distance which we call as the budget. In order to utilize the fuel more efficiently, the lab wants the UV to collect data from the best monitoring spots within one trip. In another words, the objective of path-planning problem in this scenario is to maximize the total value of the monitoring spots visited by the UV under the constraint of the fuel budget.

1.3 Literature Review

In 1987, Segev [9] first proposed the Node Weighted Steiner Tree problem, introducing non-negative vertex weights in addition to the edge weights to the problem. In his paper, one of his contributions is to introduce the Single Point Weight Steiner Tree problem, in which a special given vertex has to be included in the solution. 1n 1989, it is the first time that the term of Prize collecting Traveling Salesman is introduced by Balas [2]. Later the Prize Collecting Steiner Tree problem was first introduced by Bienstock et al. [3]. The first approximation algorithms for the Prize Collecting Traveling Salesman and Prize Collecting Steiner Tree problems with approximation guarantees of 5/2 and 3 respectively are also developed in this work. Goemans and Williamson [5,6] improved the approximation guarantee of both algorithms. Minkoff [8] proposed a modification of the algorithm for the Prize Collecting Steiner Tree problem and the primal-dual 2-approximation algorithm based on Goemans and Williamson's work. A practical heuristic denoted as the Prize-Multiplier approach is also proposed in [8]. In this thesis, we will mainly focus on implementing this Prize-Multiplier approach, which was proposed in Minkoff [8] , for our path planning problems involving ground robots.

1.4 Thesis Overview

In section 2, we build a mathematical model for the problem and formulate the quota problem and the budget problem. The formulation presented in this section will be utilized to solve the problem in CPLEX. In section 3, we present and study the heuristic algorithms we are implementing for the problem. We mainly implemented the Prize-Multiplier approach proposed by Minkoff. We conclude the thesis in section 4. We evaluate the heuristic algorithms by comparing and discussing the solutions to the problem generated by CPLEX and the heuristic algorithms. Finally, we summarize the work we have done and present the possible directions for the future research.

2. PROBLEM FORMULATION

2.1 General Model

To solve the problem, we first build a general model mathematically. We consider a graph $G(V, E)$. Let V denote the set of all vertices in the graph. Let E denote the set of all the undirected edges joining any two vertices in V. Noticing there is a root vertex serving as both depot and terminal, $o \in V$. A UV is parking at the root. It can visit the vertices in the graph by traveling along the edges belongs to E. We consider two characteristics of this undirected graph $G(V, E)$, c_e and π_v . Let c_e denote the cost of traveling through edge e. Let π_v denote the collected prize at vertex v. Let edge e_{ij} denotes the edge joins vertices i and j. We assume that the costs satisfy the triangle inequality, i.e, for $i,j,k \in V$, $c_{e_{ij}} + c_{e_{jk}} \geq c_{e_{ki}}$. Note that each vertex in the graph can be visited no more than once by the UV. Let $T(V',E')$ denote the solution tour of the UV. V' is all the vertices visited in the tour, and E' is all the edges travelled through by the UV along the tour.

(1) Quota Problem

Let Q denote the nonnegative quota. The quota constraint is that the total prize collected from the vertices visited along the tour should be no less than the quota Q. The objective of the problem is to find a tour $T(V',E')$ such that the total costs of traveling along the tour is minimized, subject to the quota constraint. That is,

$$\begin{aligned} &\min && \textstyle\sum_{e\in E'} c_e \\ &\text{subject to:} && \\ & && \textstyle\sum_{v\in V'} \pi_v \geq Q \end{aligned}$$

(2) Budget Problem

Let *B* denote the nonnegative budget. The budget constraint is that the total costs of traveling along the tour should be no greater than the budget *B*. The objective of the problem is to find a tour $T(V', E')$ such that the total prize collected from the vertices visited along the tour is maximized, subject to the budget constraint. That is,

$$\begin{array}{ll} \min & \sum_{v \in V'} \pi_v \\ \text{subject to:} & \\ & \sum_{e \in E'} c_e \leq B \end{array}$$

2.2 Quota Problem Formulation

Let *arc*(*i, j*) denote the directed arc which starts from vertex *i* and ends at vetex *j.* Let E_1 denote the set of arcs joining any two vertices in the graph. We then covert the undirected graph *G*(*V, E*) to a directed one, $G_1(V, E_1)$, by replacing each undirected edge e_{ij} with two arcs, which are *arc*(*i, j*) and *arc*(*j, i*). Accordingly, let c_{ij} denote the cost of traveling along *arc*(*i, j*) . π_v still denotes the prize collected at vertex *v*.

In this formulation, we use integer variable x_{ij} for any vertices $i, j \in V$ to decide whether *arc*(*i, j*) is chosen in the tour. Arc *arc*(*i, j*) is present in the tour if and only if $x_{ij} = 1$. Similarly, integer variable y_i for any vertex $i \in V$ decides whether vertex *i* is visited in the tour. Vertex *i* is present in the tour if and only if $y_i = 1$.

Suppose for vertex *k*, there is a virtual shipment of commodity originating from the root and then flowing along the tour. During the trip of the shipment, if and only if vertex *k* is visited, the shipment will be unloaded at vertex *k*. If vertex *k* is missed by the tour, the shipment will return to the root after finishing the trip. Under this hypothesis, we introduce a third integer variable, f_{ij}^k for any vertices $i, j \in V$ and $k \in V \setminus \{o\}$. The flow of the *k*th commodity, which is for vertex *k*, flows from vertex *i* to vertex *j* if and only if when $f_{ij}^k = 1$. This variable ensures that every vertex in the tour is reachable from the root with arcs chosen in the tour. Note that x_{ij} can then be interpreted as the capacity of *arc*(*i,j*) for the *k*th commodity, and y_k can be interpreted as the demand of the *k*th commodity at vertex *k*.

The mathematical expression of these 3 integer variables is as below:

$$x_{ij} = \begin{cases} 1 & \text{if } arc\ (i,j) \in E_1 \text{ is in the tour} \\ 0 & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if vertex } i \text{ is visited in the tour} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{ij}^k = \begin{cases} 1 & \text{if the flow of the } k\text{th commodity flows from vertex } i \text{ to vertex } j \\ 0 & \text{otherwise} \end{cases}$$

The objective function of quota problem is quite similar to what we've discussed in section 2.2, which is

$$\min \qquad \sum_{i,j\in V} c_{ij} x_{ij}$$

The quota constraint can be written as:

$$\sum_{i\in V} \pi_i y_i \geq Q$$

As we were noticing in the previous part of this section, x_{ij} can then be interpreted as the capacity of *arc*(*i,j*) for the *k*th commodity. Thus we get the capacity constraint as,

$$f_{ij}^k \leq x_{ij} \qquad \forall i,j \in V, \forall k \in V \setminus \{o\}$$

To make sure that the shipment of *k*th commodity satisfies the demand of the *k*th target, and can only be unloaded at vertex *k*, we formulate a set of 3 flow balance constraints. The first one is,

$$\sum_{j\in V\setminus\{v_0\}} (f_{oj}^k - f_{jo}^k) = y_k \qquad \forall k \in V \setminus \{o\} \text{ and } o \text{ is the root}$$

This constraint ensures that the flow of the *k*th commodity comes out of the root *o* has to satisfy the demand of the *k*th vertex before it comes back to the root. The second flow balance constraint is,

$$\sum_{j\in V} (f_{jk}^k - f_{kj}^k) = y_k \qquad \forall k \in V \setminus \{o\}$$

This constraint ensures that the flow of the *k*th commodity comes into vertex *k* should satisfy the demand of the *k*th vertex before it comes out. The last flow balance constraint is,

$$\sum_{j\in V} f_{ij}^{k} = \sum_{j\in V} f_{ji}^{k} \qquad \forall i,k \in V \setminus \{o\} \text{ and } i \neq k$$

This constraint ensures that the *k*th commodity is not unloaded at any vertex *i* when $i \neq k$.

To avoid the tour visiting the same vertex repeatedly and forming a circle, which contains only one vertex, we add this constraint,

$$x_{jj} = 0 \qquad \forall j \in V$$

To ensure that the solution tour starts and ends at the same vertex, we add a constraint which restricts the in-degree to be equal to the out-degree for each vertex in the graph except the root. Here is the constraint,

$$\sum_{i\in V} (x_{ij} - x_{ji}) = 0 \qquad \forall j \in V \setminus \{o\}$$

The last constraint makes sure that there's only one arc out of root *o*.

$$\sum_{j\in V\setminus\{o\}} x_{oj} = 1$$

Now we get the integer program for the quota problem, which is as below,

$$\max \quad \sum_{i,j \in V} c_{ij} x_{ij}$$

subject to:

$$\sum_{i \in V} \pi_i y_i \geq Q$$

$$f_{ij}^k \leq x_{ij} \qquad \forall i, j \in V, \forall k \in V \setminus \{o\}$$

$$\sum_{j \in V \setminus \{o\}} (f_{oj}^k - f_{jo}^k) = y_k \qquad \forall k \in V \setminus \{o\}$$

$$\sum_{j \in V} f_{ij}^k = \sum_{j \in V} f_{ji}^k \qquad \forall i, k \in V \setminus \{o\} \text{ and } i \neq k$$

$$\sum_{j \in V} (f_{jk}^k - f_{kj}^k) = y_k \qquad \forall k \in V \setminus \{o\}$$

$$x_{jj} = 0 \qquad \forall j \in V$$

$$\sum_{i \in V} (x_{ij} - x_{ji}) = 0 \qquad \forall j \in V \setminus \{o\}$$

$$\sum_{j \in V \setminus \{o\}} x_{oj} = 1$$

$$x_{ij}, y_i, f_{ij}^k \in \{0,1\} \qquad \forall i, j, k \in V \text{ and } k \neq o$$

2.3 Budget Problem Formulation

For the budget problem, we use the same integer variables as we've used for the quota problem in section 2.2, which are x_{ij}, y_i, f_{ij}^k for any $i, j \in V$ and $k \in V \setminus \{o\}$.

The objective function of the budget problem is,

$$\max \quad \sum_{i \in V} \pi_i y_i$$

The budget constraint can be written as,

$$\sum_{i,j\in V} c_{ij}x_{ij} \le B$$

Obviously, the rest of the constraints, such as the capacity constraint, the flow balance constraints and other constraint, are the same with the quota problem. We get the integer program of the budget problem, which is as below,

$$\max \quad \sum_{i\in V} \pi_i y_i$$

subject to:

$$\sum_{i,j\in V} c_{ij}x_{ij} \le B$$

$$0 \le f_{ij}^k \le x_{ij} \qquad \forall i,j \in V, \forall k \in V \setminus \{o\}$$

$$\sum_{j\in V\setminus\{o\}} (f_{oj}^k - f_{jo}^k) = y_k \qquad \forall k \in V \setminus \{o\}$$

$$\sum_{j\in V} f_{ij}^k = \sum_{j\in V} f_{ji}^k \qquad \forall i,k \in V \setminus \{o\} \text{ and } i \ne k$$

$$\sum_{j\in V} (f_{jk}^k - f_{kj}^k) = y_k \qquad \forall k \in V \setminus \{o\}$$

$$x_{jj} = 0 \qquad \forall j \in V$$

$$\sum_{i\in V} (x_{ij} - x_{ji}) = 0 \qquad \forall j \in V \setminus \{o\}$$

$$\sum_{j\in V\setminus\{o\}} x_{oj} = 1$$

$$x_{ij}, y_i, f_{ij}^k \in \{0,1\} \qquad \forall i,j,k \in V \text{ and } k \ne o$$

3. HEURISTICS

3.1 General Ideal of Implementing The Heuristic Algorithm

The goal of our thesis is to implement a heuristic algorithm to solve the quota problem and the budget problem. We expect performance of the heuristic algorithm with better time efficiency and acceptable approximate result.

The general plan of implementing the heuristic algorithm is as below:

(1) Formulate the problem base on a mathematical model similar to the Prize Collecting Steiner Tree (PCST) problem.

(2) Generate an approximate solution tree *T* using the prize-multiplier approach.

(3) Generate the solution tour with all the vertices spanned by tree *T*, using the LKH algorithm.

3.2 Introduction to The Prize Collection Steiner Tree Problem

Let *G*(*V, E*) denote an undirected graph with root *o*. *V* denotes the set of all the vertices in the graph. *E* denotes the set of all the edges joining any two of the vertices. c_e is the cost of traveling through edge *e*. π_v is the penalty for not visiting vertex *v*. Let e_{ij} denote the edge joining vertices *i* and *j*. We assume costs satisfy the triangle inequality, i.e. $c_{e_{ij}} + c_{e_{jk}} \geq c_{e_{ki}}$ for any vertices $i, j, k \in V$. The objective of PCST problem is to find a tree $T(V_T, E_T)$ spanning a subset of the vertices and the root vertex *o*, such that the sum of the total costs of edges in the tree and the total penalties associated with vertices not spanned by the tree is minimized.

We use the following integer variables:

$$x_e = \begin{cases} 1 & \text{if } e \text{ is in the tree} \\ 0 & \text{otherwise} \end{cases}$$

$$z_N = \begin{cases} 1 & \text{if N is the set of all vertices not spanned by the tree} \\ 0 & \text{otherwise} \end{cases}$$

Noting that subset of vertices $N \subseteq V \setminus \{o\}$.

As we stated above, the objective function of PCST problem is

$$\min \quad \sum_{e \in E} c_e x_e + \sum_{N \subseteq V \setminus \{o\}} z_N \sum_{i \in N} \pi_i$$

Let S denote a subset of vertices not containing the root vertex *o*. If subset *S* only contains vertices spanned by solution tree *T*, obviously *S* and T intersect each other. Thus there must be at least on edge of *T* crossing the outer edge of subset *S*. If subset *S* contains not only the vertices spanned by tree *T* but also the vertices out of *T*, *S* and *T* still intersect each other. This situation is the same as the previous one. If subset *S* only contains the vertices not spanned by tree *T*, this means that there is no edge of *T* crossing the outer edge of subset *S* and subset *S* is within the subset *N* that only contains all the vertices not spanned by tree *T*. We use the following expression to describe this constraint:

$$\sum_{e\in\delta(S)} x_e + \sum_{N\supseteq S} z_N \geq 1 \qquad S \subseteq V\setminus\{o\}$$

To ensure that there is at most one subset *N* that satisfies $z_N = 1$, we add a second constraint:

$$\sum_{N\subseteq V\setminus\{o\}} z_N \leq 1$$

Noting that because of the second term of the objective function, the last constraint is automatically satisfied by optimal solution. Thus we can drop it without affecting the optimization. Relaxing the constraints of integer variables, we get the linear program of PCST problem as below:

$$\begin{array}{lll}
\min & \sum_{e\in E} c_e x_e + \sum_{N\subseteq V\setminus\{o\}} z_N \sum_{i\in N} \pi_i & \\
\text{subject to:} & & \\
& \sum_{e\in\delta(S)} x_e + \sum_{N\supseteq S} z_N \geq 1 & S\subseteq V\setminus\{o\} \\
& x_e \geq 0 & e\in E \\
& z_N \geq 0 & N\subseteq V\setminus\{o\}
\end{array}$$

Corresponding to the first constraint of the above linear program, we set up a non-negative dual variable y_S for each subset of vertices $S\subseteq V\setminus\{o\}$. The dual of the linear program of PCST problem is as below:

$$\max \quad \sum_{S \subseteq V \setminus \{o\}} y_S$$

subject to:

$$\sum_{S: e \in \delta(S)} y_S \le c_e \qquad e \in E$$

$$\sum_{S \subseteq N} y_S \le \sum_{i \in N} \pi_i \qquad N \subseteq V \setminus \{o\}$$

$$y_S \ge 0 \qquad S \subseteq V \setminus \{o\}$$

3.3 Prize-Multiplier Approach for Quota Problem

The prize-multiplier approach is proposed by Minkoff in [8]. The idea of the prize-multiplier approach came out of this scenario. With the cost c_e and the prize π_v, we only want to include the vertex *v* in the solution tree if the cost of reaching the vertex is outweighed by the value of prize collected. In the situation where the solution tree does not contain enough prizes, a prize multiplier can be introduced to scale the prize of each vertex and makes the total prize in the solution tree sufficient enough to offset more cost. Thus the solution tree is forced to span more vertices. Ideally, we can obtain the desired value of prizes collected in the solution tree by adjusting the value of prize multiplier.

3.3.1 Quota Problem Formulation

As we've stated in the general plan for solving the problem in section 3.1, the objective is to generate a solution tree instead of a tour before we could apply the LKH algorithm. Thus, at this step, the objective of the new quota problem is to find a solution tree

$T(V_T, E_T)$ that total cost of edges in the tree is minimized while the total prize collected at the vertices spanned by the tree is no less than quota Q.

We use the same set of integer variables as we've used for PCST problem formulation.

$$x_e = \begin{cases} 1 & \text{if } e \text{ is in the tree} \\ 0 & \text{otherwise} \end{cases}$$

$$z_N = \begin{cases} 1 & \text{if N is the set of all vertices not spanned by the tree} \\ 0 & \text{otherwise} \end{cases}$$

The objective function is

$$\min \quad \sum_{e \in E} c_e x_e$$

The first two constraints are the same ones as for PCST problem,

$$\sum_{e \in \delta(S)} x_e + \sum_{N \supseteq S} z_N \geq 1 \quad S \subseteq V \setminus \{o\}$$

$$\sum_{N \subseteq V \setminus \{o\}} z_N \leq 1$$

The third constraint is the quota constraint. In the previous section, we've written this constraint as below,

$$\sum_{i \in V_T} \pi_i \geq Q$$

We intend to introduce the same integer variables we've used for PCST problem formulation in section 3.1 into the quota constrain. Thus we have to make modifications to the above expression. First we get

$$\sum_{i \in V} \pi_i - \sum_{i \in V_T} \pi_i \le \sum_{i \in V} \pi_i - Q$$

Noting that the left hand side of the above inequality is the total prize of all vertices in the graph minus the total prize collected at the vertices spanned by the solution tree *T*. In another word, the left hand side is the total prize of all the vertices not spanned by the solution tree *T*. Thus we can re-write the quota constraint as below,

$$\sum_{N \subseteq V \setminus \{o\}} z_N \sum_{i \in N} \pi_i \le \sum_{i \in V} \pi_i - Q$$

Relaxing the constraints of the integer variables and dropping the second constraint, we get the linear programming relaxation to the new quota problem as below,

$$\begin{array}{lll} \min & \sum_{e \in E} c_e x_e & \\ \text{subject to:} & & \\ & \sum_{e \in \delta(S)} x_e + \sum_{N \supseteq S} z_N \ge 1 & S \subseteq V \setminus \{o\} \\ & \sum_{N \subseteq V \setminus \{o\}} z_N \sum_{i \in N} \pi_i \le \sum_{i \in V} \pi_i - Q & \\ & x_e, z_N \ge 0 & N \subseteq V \setminus \{o\} \end{array}$$

Non-negative dual variable y_S for each subset of vertices $S \subseteq V \setminus \{o\}$ is corresponding to the first constraint. A new non-negative dual variable μ is corresponding to the second constraint. We write the dual of this linear program as below,

$$\max \quad \sum_{S\subseteq V\setminus\{o\}} y_S - \mu\Big(\sum_{i\in V}\pi_i - Q\Big)$$

subject to:

$$\sum_{S:e\in\delta(S)} y_S \le c_e \qquad e\in E$$

$$\sum_{S\subseteq N} y_S \le \sum_{i\in N}\mu\pi_i \qquad N\subseteq V-r$$

$$\mu, y_S \ge 0 \qquad S\subseteq V-r$$

By observing the objective function of the dual program, we can easily notice that when μ is fixed, the second term of the objective function is constant. For each fixed μ, this term will not affect the optimization. Comparing the constraints of the above dual program with the ones of the dual program for PCST problem, the only difference is that in the second constraint of the above dual program, there is μ serving as a prize multiplier. Thus solving the dual program for the new quota problem is identical to solving the dual program for PCST problem with prizes scaled by μ.

To solve the quota problem with the prize-multiplier approach, we first choose a set of μ. Then we run a primal-dual algorithm on the PCST problem once for each chosen μ. Among the set of solution trees we have generated, we pick a tree that satisfy the quota requirement and has a total prize collected as close to the quota Q as possible.

3.3.2 Bisection Search

In order to reach the desired solution tree in a shorter time, we apply bisection method to search the appropriate prize-multiplier. Let $[\mu_{lo}, \mu_{hi}]$ be the search interval. At each step, compute the midpoint of the interval $\mu_{mid} = (\mu_{lo} + \mu_{hi}) / 2$. The search interval is divided in two now, which are $[\mu_{lo}, \mu_{mid}]$ and $[\mu_{mid}, \mu_{hi}]$. Then we generate a solution tree *T* with prize-multiplier μ_{mid}. Compare the total prize collected in tree *T*, $\prod_T$, with the quota Q. If $\prod_T \geq Q$, the solution tree *T* is the best feasible solution we have obtained until this step. We record this solution and then try to obtain a new tree T' with lower total prize $\prod_{T'}$ using a smaller prize-multiplier μ. Following the principle of bisection search, we start a new bisection search in the subinterval $[\mu_{lo}, \mu_{mid}]$. If $\prod_T < Q$, we need to increase the prize-multiplier μ to obtain a solution tree T' with greater total prize $\prod_{T'}$. Thus we start a new bisection search in the subinterval $[\mu_{mid}, \mu_{hi}]$. Repeat the steps until the best solution obtained in the process cannot be improved much more. The best solution is the last feasible solution obtained in the bisection steps. Record the best solution for further procedure.

The algorithm we deployed to solve the PCST problem in each bisection search step is a primal-dual algorithm developed by Bae et al.[1]. This algorithm is essentially developed for a two-depot heterogeneous traveling salesman problem. We implement it for our single depot PCST problem by intentionally setting the cost of each edge joining any two vertices except the depot as infinity for the second vehicle. The cost of each edge joining the depot for the second vehicle and any other vertex *v* is set to be the prize collected at the vertex *v* by the first vehicle. The input data for the first vehicle are the data for our original PCST problem. After running the algorithm, the result tree for the first vehicle is our solution tree. The result for the second vehicle is ignored.

3.3.3 Pruning Heuristic for The Quota Problem

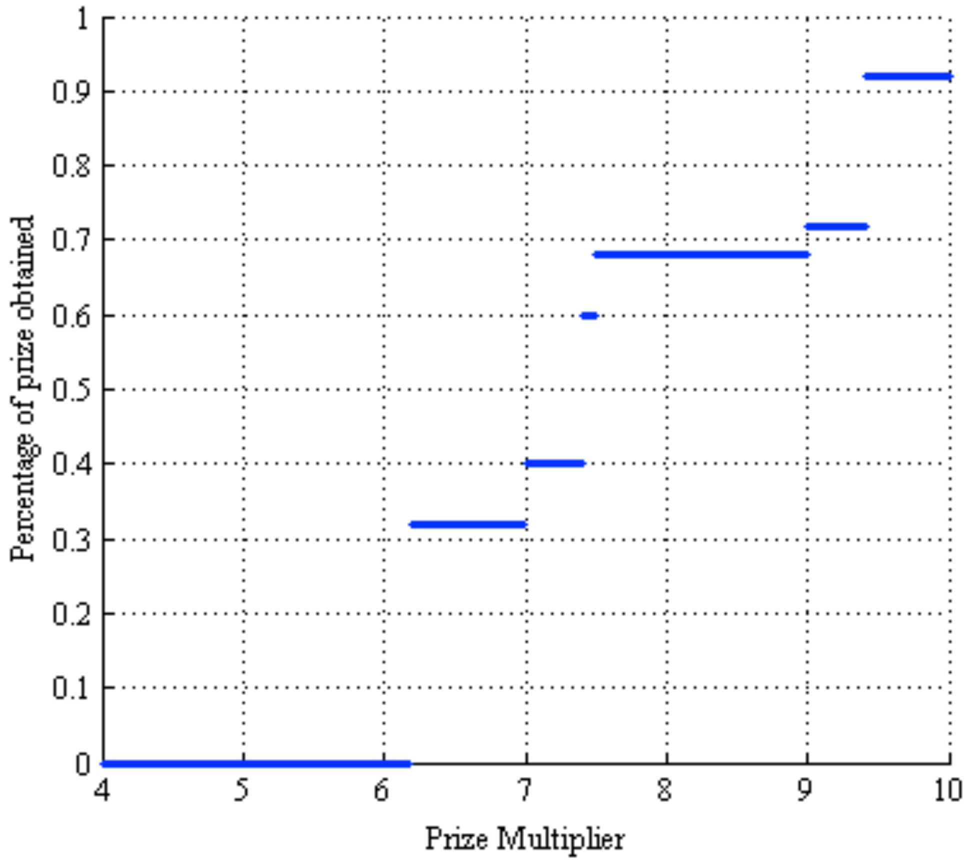

Figure 3-1: Percentage of prize obtained with prize-multiplier approach

It is always possible to obtain a solution tree with total prize to be no less then the given quota Q when Q is smaller than the total prize of all vertices in the graph. By choosing a proper bisection search interval, a best solution out of the prize-multiplier approach can be achieved. We hope by performing more iterations of bisection search, the total prize collected in the solution tree will be approaching the quota Q gradually and getting closer and closer to Q. When the total prize reaches Q or arrives at close enough to Q, a solution tree with decent approximating rate is obtained. However this is not the case. As we can see from the Figure 3-1, when we run the algorithm with prize-multiplier μ increasing gradually and slowly by the step of 0.01, we get a set of solution trees with total prize increasing discretely.

Given a very small $\varepsilon > 0$, let $\mu_2 = (1+\varepsilon)\mu_1$. Let T_1 denote to solution tree generated with prize-multiplier μ_1 and $\prod_{T_1}$ denote the total prize of tree T_1. Let T_2 denote to solution tree corresponding to prize-multiplier μ_2 and $\prod_{T_2}$ denote the total prize of tree T_2, $\prod_{T_1} < \prod_{T_2}$. Suppose the given quota *Q* falls between $\prod_{T_1}$ and $\prod_{T_2}$, $\prod_{T_1} < Q < \prod_{T_2}$. It is possible that within the feasible solution tree T_2 there exists a subtree T_3 whose total edge cost is smaller and yet a feasible solution. The goal of pruning heuristic for the quota problem is to prune away vertices that decrease the total edge cost as much as possible while still maintain a total prize no less than quota *Q*. To conduct the pruning procedure more effective and efficient, we always try to prune away subtrees with more edge cost and less prize. Thus we compute the cost-prize ratio of each subtree. This ratio denotes the value of edge cost per unit of prize of a subtree.

Given a solution tree *T* and a non-negative quota Q, suppose vertex *u* and *v* are visited in tree *T*. Let T_u denote the subtree within tree *T* and rooted at vertex *u*. The edge e_{uv} is in tree T and joins vertex *u* and *v*. Vertex *v* is not spanned by subtree T_u. We call vertex *v* the parent of vertex *u*. Let c_{uv} denote the cost of edge e_{uv}, and C_{T_u} denote the total cost of edges in subtree T_u. Let $\prod_{T_u}$ denote the total cost collected in subtree T_u. The cost-prize ratio of subtree T_u is defined as $R_u = \frac{C_{T_u} + c_{vu}}{\prod_{T_u}}$.

The detailed pruning steps are as below:

```
Quota-Prune(T, Q)

1  sort all subtrees $T_u$ on the decreasing order of cost-prize ratio $R_u$
2  **For** each $T_u$ in decreasing order of $R_u$ **do**
   **if** $\prod_T - \prod_{T_u} \geq Q$ **then**
       $T \leftarrow T - T_u$; break,
   **end if**
   **end for**
3  update $R_v$ for each vertex *v* on the path *u* ~ root
```

Repeat the above pruning steps until there is no subtree can be removed without leaving the remained subtree unsatisfying the quota requirement.

Run LKH algorithm using the result tree from pruning steps as input to generate the final solution tour for the Quota Problem. LKH was developed by Helsgaun [6]. It is one of the best heuristic for single vehicle Traveling Salesman Problem. We use the executable package downloaded online at [7].

3.3.4 Summary

To sum up, the steps of solving the quota problem with the prize-multiplier approach are listed below

Steps of solving the quota problem with heuristic

- run primal-dual algorithm for PCST problem with prize-multiplier μ and obtain a solution tree T_μ
- repeat the above step, using bisection search to find a prize-multiplier μ, such that the corresponding solution tree T_μ is feasible and has a total prize as close to the quota Q as possible
- prune the chosen tree T_μ to get a new feasible tree T_μ'
- run LKH algorithm on tree T_μ' to generate the final solution tour

3.4 Prize-Multiplier Approach for Budget Problem

According to the dual program to the quota problem we discussed in previous section, each time we solve the dual for with a fixed prize-multiplier μ, we get a solution tree, which is independent of the quota Q.

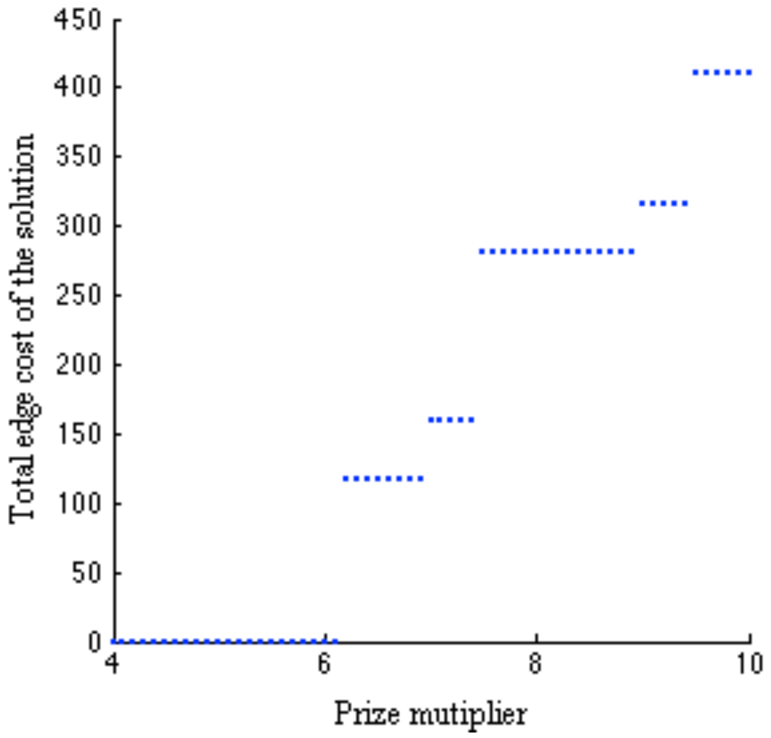

Figure 3-2: Total edge cost of the solution

As we increasing the prize-multiplier μ, we are forcing the solution tree to include more vertices with prize. In the mean time, the solution is also forced to add more edges with cost, as we can see from Figure 3-2. Thus the idea of the prize-multiplier approach for budget problem is to increase prize-multiplier μ in order to maximize the total prize collected in the tree until the total edge cost reaches the budget *B*.

3.4.1 Bisection Search

Let C_{tree} denote the total edge cost of a tree. Let C_{tour} denote the total edge cost of the tour generated out of all the vertices in the tree. Obviously, $C_{tour} \neq C_{tree}$. In another word, it is possible that the tree satisfies the budget requirement while the tour does not. Thus, we run LKH algorithm on the solution tree in each iteration of bisection search and perform the search using the total edge cost of solution tour.

Let $[\mu_{lo}, \mu_{hi}]$ be the search interval. At each step, compute the midpoint of the interval $\mu_{mid} = (\mu_{lo} + \mu_{hi}) / 2$. The search interval is divided in two now, which are $[\mu_{lo}, \mu_{mid}]$ and $[\mu_{mid}, \mu_{hi}]$. Then we generate a solution tour *T* with prize-multiplier μ_{mid}. Compare the total edge cost in tour *T*, C_T, with the given budget B. If $C_T \leq B$, the solution tour *T* is the best feasible solution we have obtained until this step. We record this solution and then try to obtain a new tour T' with more total prize $\prod_{T'}$ using a larger prize-multiplier μ. Following the principle of bisection search, we start a new bisection search in the subinterval $[\mu_{mid}, \mu_{hi}]$. If $C_T > B$, the solution tour *T* contains the least edge cost among all the infeasible solutions we have obtained until this step. We call it the best infeasible solution. We need to decrease the prize-multiplier μ to obtain a solution tour T' with less total prize $\prod_{T'}$. Thus we start a new bisection search in the subinterval $[\mu_{lo}, \mu_{mid}]$. Repeat the steps and keep the record of both the best feasible solution tree and the best infeasible solution tree, until the best infeasible solution obtained in the process cannot be improved much more without bring the solution to be feasible. We use the best infeasible solution tour as the original tour for pruning steps. After the pruning step, we choose a solution tour with more total prize as the final solution between the best feasible solution our we obtained after the bisection search and the resulting subtour of the pruning steps.

3.4.2 Pruning Heuristic for The Budget Problem

Given a solution tour *T* and a non-negative budget B, suppose vertex *p, q, u* and *v* are visited in tour *T*. Directed arcs *arc*(*p, u*) and *arc*(*v, q*) are in tour *T*. Let T_{uv} denote a segment of tour *T* that is rooted at vertex *u* and ends at vertex *v*. Arc *arc*(*p, q*) is not in tour *T* and joins vertex *p* and *q*. Let c_{ij} denote the cost of arc *arc*(*i, j*) for any two vertices in the tour, and $C_{T_{uv}}$ denote the total cost of edges in T_{uv}. Let $\Pi_{T_{uv}}$ denote the total cost collected in subtree T_u. When T_{uv} is removed from *T*, the total cost of removed edge is as below

$$C(u,v) = C_{T_{uv}} + c_{pu} + c_{vq}$$

The cost-prize ratio of T_{uv} is defined as

$$R_u = \frac{C(u,v) - c_{pq}}{\Pi_{T_{uv}}}$$

```
Budget-Prune($T$, $B$)

1 sort all $T_{uv}$ on $R_{uv}$ in decreasing order
2 count = 0
3 for each $T_{uv}$ in decreasing order of $R_{uv}$
4       if $C_T - C(v,u) + c_{pq} > B$ then
5           $T \leftarrow T - T_{uv} + e_{pq}$; break cycle
6       else count = count + 1
5       end if
6 end for
7 if count = number of vertices in $T_{uv}$
8     find the first $T_{uv}^{\max}$ with the smallest $\prod_{T_{uv}}$
9     do $T \leftarrow T - T_{vu}^{\max} + e_{pq}$
9 end if
10 update $R_{uv}$, $\prod_{T_{uv}}$ and $T_{uv}$ for new $T$
```

Repeat the above pruning steps until the result tour is feasible. Compare the result tour with the best feasible solution we have obtained in previous bisection search steps, choose the one with greater total prize as the final solution of the heuristic algorithm for the budget problem.

3.4.3 Summary

To sum up, the steps of solving the budget problem with the prize-multiplier approach are listed below

---

Steps of solving the budget problem with heuristic

- run primal-dual algorithm for PCST problem with prize-multiplier μ and obtain a solution tree T_μ
- apply LKH algorithm on T_μ to generate a tour T_μ'
- repeat the above steps, using bisection search to find prize-multipliers μ_1 and μ_2, such that the corresponding tour T_{μ_1}' is infeasible and has a total edge cost as close to the budget *B* as possible, and tour T_{μ_2}' is feasible and has a total edge cost as close to the budget *B* as possible.
- prune away vertices in tour T_{μ_1}' until the resulting tour T_{μ_1}'' is feasible
- pick the tour with greater total prize between T_{μ_2}' and T_{μ_1}'' as the final solution

---

4. SIMULATION RESULTS

4.1 Input Data to The Quota Problem and The Budget Problem

We solve the quota problem and budget problem on two different UV models. The first model of UV can move in any direction, and has no constraint on the minimum turning radius. We call this UV model the simple model. The second model we implement is the famous Reeds-Shepp model. A Reeds-Shepp UV can move both forwards and backwards and can turn fully left and fully right. There is a constraint on the minimum turning radius for the Reeds-Shepp UV.

Initially we generate a number of vertices randomly distributing in a test area of 1000m^2. The traveling distance between any two vertices in the graph for UV is used as the cost of edge joining the two vertices. The prize collected at each vertex is set to be 1. Let *n* denote the number of vertices in the test area. Set the quota $Q = \frac{n}{2}$. Set the budget $B = 1500, 1800, 2000, 2500, 2800$ for $n = 10, 20, 30, 40, 50$ correspondingly.

Let *n* denote the number of vertices in the test area. We run the algorithms with input data of n=10 (20, 30, 40, 50) for 50 instance respectively. In order to evaluate the solution quality of the heuristic algorithm, we solve the problem in CPLEX using the formulation we developed in section 2 and the same input data.

The codes for the heuristic is written in MATLAB and run on an Intel Core i7 2.9Ghz/8GB laptop. The codes for CPLEX is written in C++ and run on an AMD Athlon(tm) II X4 640 3.0Ghz/8GB desktop.

4.2 Result and Quality Analysis

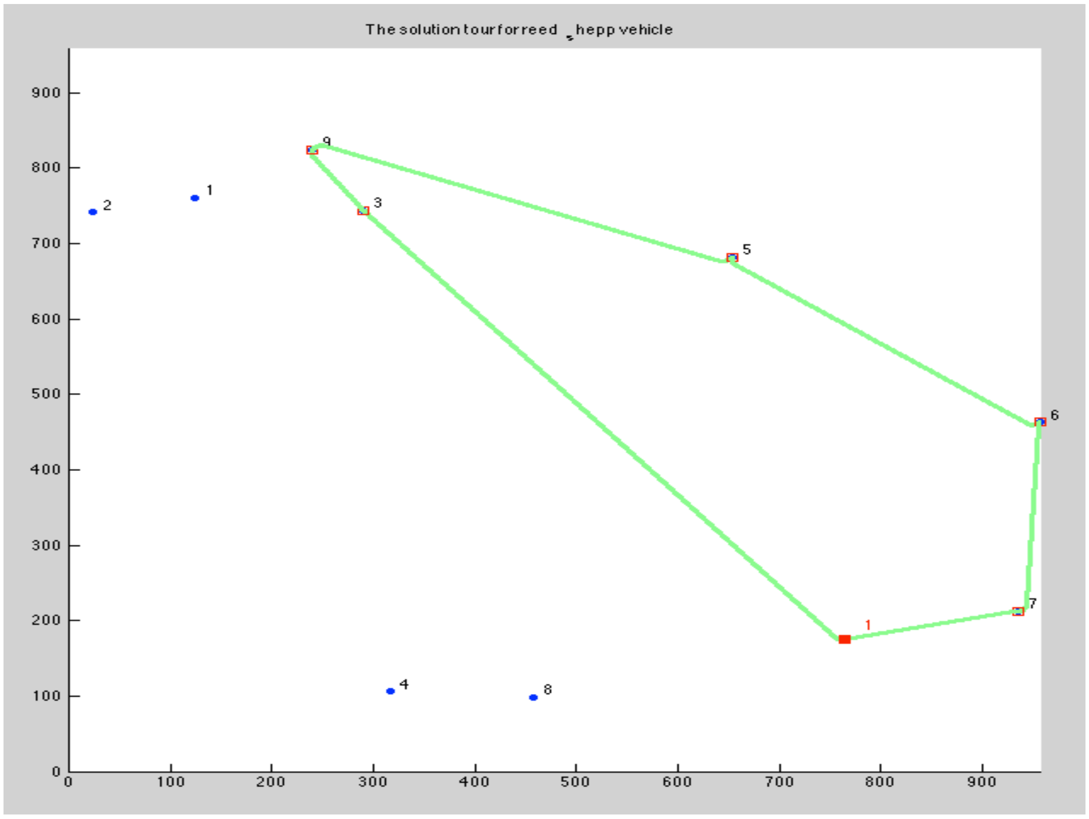

Figure 4-1: An example of solution to the quota problem with Reeds-Shepp UV

In Figure 4-1, an approximate Reeds-Shepp tour is generated for *n*=10 and *Q*=5. We can see that the total prize in the tour is 5. This is a feasible solution.

Each time we get a result from the algorithms, we record the running time for optimization and the objective value. Let t_{CPLEX} denote the average running time of optimization for each value of *n* in CPLEX. Let $t_{heuristc}$ denote the average running time of the heuristic algorithm for each value of *n*. Let C_{CPLEX} and Π_{CPLEX} denote the total edge cost and total prize of the solution tour generated by CPLEX. Let $C_{heuristic}$ and $\Pi_{heuristic}$ denote the total edge cost and total prize of the solution tour generated from the heuristic algorithm. Define the solution quality for the quota problem as below:

$$SQ_{quota} = \frac{C_{heuristic} - C_{CPLEX}}{C_{heuristic}} \times 100\%$$

The solution quality for the budget problem is defined as below

$$SQ_{budget} = \frac{\Pi_{CPLEX} - \Pi_{heuristic}}{\Pi_{CPLEX}} \times 100\%$$

We also compare $t_{heuristc}$ with t_{CPLEX} to analyze the time efficiency of the heuristic algorithm.

Table 4-1: The results of the quota problem for the simple vehicle model

Number of Vertices	Average Solution Quality	Average Time/sec	
		CPLEX	Heuristic
10	9.50%	0.64	1.59
20	11.36%	33.72	4.69
30	8.63%	267.58	7.96
40	9.63%	1937.09	16.84
50	12.34%	6000.31	37.23

Table 4-2: The results of the quota problem for the Reeds-Shepp vehicle model

Number of Vertices	Average Solution Quality	Average Time/sec	
		CPLEX	Heuristic
10	5.07%	0.70	1.08
20	10.55%	23.97	3.09
30	14.45%	414.16	7.15
40	15.30%	2803.31	14.75
50	16.11%	6815.03	24.95

Table 4-3: The results of the budget problem for the simple vehicle model

Number of Vertices	Average Solution Quality	Average Time/sec	
		CPLEX	Heuristic
10	9.64%	0.55	1.53
20	10.54%	24.20	3.88
30	11.82%	217.48	8.13
40	11.67%	962.53	15.79
50	12.98%	5330.72	24.89

Table 4-4: The results of the budget problem for the Reeds-Shepp vehicle model

Number of Vertices	Average Solution Quality	Average Time/sec	
		CPLEX	Heuristic
10	7.27%	0.70	1.20
20	13.43%	28.32	5.23
30	15.59%	365.56	7.83
40	15.04%	716.84	13.22
50	15.17%	3457.70	24.35

Observing the data in Table 4-1,4-2,4-3,4-4, we come to these conclusions. By comparing with the results generated by CPLEX, we note that the value of solution quality of heuristic algorithm increases with *n*. This means that the heuristic algorithm produces better approximate solution when n is small. When *n* is 50, the average solution quality is within the range of [12%, 17%]. We also notice that the heuristic algorithm performs slightly better with the simple vehicle model than with the Reeds-Shepp vehicle. However the heuristic algorithm has a significant advantage in time efficiency as *n* grows larger. When *n* is 10, CPLEX wins in the running time comparison. As n grows from 20 to 50, the running time of CPLEX increases exponentially while the running time of the heuristic algorithm stays at an acceptable range. When *n* reaches 50, the running time of CPLEX is almost 200 times of the one of the heuristic algorithm. Thus we can conclude that the heuristic algorithm we implemented generates acceptable approximate rate with significant improvement in time efficiency.

5. CONCLUSIONS AND FUTURE WORK

In this thesis we implemented a heuristic algorithm of prize-multiplier approach on the single vehicle quota problem and budget problem. We tested the algorithms two different models of vehicle. This heuristic performs well when solving our problems with both vehicle models. It provides a solution with good approximation rate in a small period of time. The quality of the solutions obtained with the heuristic algorithm drops as the number of vertices increases. The solution produced by CPLEX using the multy-commodity flow formulation has better quality but is very time consuming as the number of vertices increases.

In future work, the heuristic can be tested on new vehicle models with asymmetric edge cost matrixes and more generalized vertex prize matrixes. Also, the solutions from the heuristic can be improved trough the framework of improvement heuristics.

REFERENCES

[1] Bae, Jungyun and Sivakumar Rathinam. "A Primal Dual Algorithm for a Heterogeneous Traveling Salesman Problem." *arXiv preprint arXiv:1111.0567* (Nov, 2011).

[2] Balas, E. "The Prize Collecting Traveling Salesman Problem." *Networks* 19, no. 6 (1989): 621-636.

[3] Bienstock, Daniel, Michel X. Goemans, David Simchi-Levi, and David Williamson. "A Note on the Prize Collecting Traveling Salesman Problem." *Mathematical Programming* 59, no. 1-3 (mar, 1993): 413-420.

[4] Garg, N. "A 3-Approximation for the Minimum Tree Spanning k Vertices." *Proceedings of the 37th Annual IEEE Symposium on Foundation of Computer Science,* (1996): 302-309.

[5] Goemans, M. X. and D. P. Williamson. "A General Approximation Technique for Constrained Forest Problems." *SIAM Journal on Computing* 24, no. 2 (1995): 296-317.

[6] Goemans, M. X. and D. P. Williamson. "The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems." *Approximation Algorithms for NP-Hard Problems* (1997): 144-191.

[7] Helsgaun, Keld. "Lin-Kernighan Heuristic." Retrieved May 6, 2013, from http://www.akira.ruc.dk/~keld/research/LKH/.

[8] Lin, S. and B. W. Kernighan. "An Effective Heuristic Algorithm for the Traveling-Salesman Problem." *Operations Research* vol. 21 (1973): 498-516.

[9] Minkoff, Maria. "The Prize Collecting Steiner Tree Problem."Massachusetts Institute of Technology, 2000.

[10] Segev, Arie. "The Node-Weighted Steiner Tree Problem." *Networks* 17, no. 1 (1987): 1-17.

[11] "Packbot." In *Wikipedia*. Retrieved May 6, 2013, from http://en.wikipedia.org/wiki/PackBot.