ALGORITHMS FOR MULTIPLE VEHICLE ROUTING PROBLEMS

A Dissertation

by

JUNG YUN BAE

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

DOCTOR OF PHILOSOPHY

Chair of Committee,	Sivakumar Rathinam
Committee Members,	Swaroop Darbha
	Reza Langari
	Sergiy Butenko
Head of Department,	Andreas Polycarpou

May 2014

Major Subject: Mechanical Engineering

ABSTRACT

Surveillance and monitoring applications require a collection of heterogeneous vehicles to visit a set of targets. This dissertation considers three fundamental routing problems involving multiple vehicles that arise in these applications. The main objective of this dissertation is to develop novel approximation algorithms for these routing problems that find feasible solutions and also provide a bound on the quality of the solutions produced by the algorithms.

The first routing problem considered is a multiple depot, multiple terminal, Hamiltonian Path problem. Given multiple vehicles starting at distinct depots, a set of targets and terminal locations, the objective of this problem is to find a vertex-disjoint path for each vehicle such that each target is visited once by a vehicle, the paths end at the terminals and the sum of the distances travelled by the vehicles is a minimum. A 2-approximation algorithm is presented for this routing problem when the costs are symmetric and satisfy the triangle inequality. For the case where all the vehicles start from the same depot, a $\frac{5}{3}$-approximation algorithm is developed.

The second routing problem addressed in this dissertation is a multiple depot, heterogeneous traveling salesman problem. The objective of this problem is to find a tour for each vehicle such that each of the targets is visited at least once by a vehicle and the sum of the distances travelled by the vehicles is minimized. A primal-dual algorithm with an approximation ratio of 2 is presented for this problem when the vehicles involved are ground vehicles that can move forwards and backwards with a constraint on their minimum turning radius.

Finally, this dissertation addresses a multiple depot heterogeneous traveling salesman problem when the travel costs are asymmetric and satisfy the triangle inequality.

An approximation algorithm and a heuristic is developed for this problem with simulation results that corroborate the performance of the proposed algorithms. All the main algorithms presented in the dissertation advance the state of art in the area of approximation algorithms for multiple vehicle routing problems.

This dissertation has its value for providing approximation algorithms for the routing problems that involves multiple vehicles with additional constraints. Some algorithms have constant approximation factor, which is very useful in the application but difficult to find. In addition to the approximation algorithms, some heuristic algorithms were also proposed to improve solution qualities or computation time.

ACKNOWLEDGEMENTS

It was such an honor for me to complete my Ph.D degree in the Department of Mechanical Engineering at Texas A&M University. I would like to thank the people who helped me during all those years.

My first thanks go to my advisor, Dr. Sivakumar Rathinam. It was through his support and encouragement that I nurtured my interest in routing problems and unmanned vehicle systems. At every challenging moment of a mental block, Dr. Rathinam helped me stay motivated and try another direction by giving me insightful advice with patience. His detailed guidance made me a better researcher, teacher and presenter. I also thank Dr. Swaroop Darbha, who in all practical terms served as my co-advisor throughout the process. I owe an enormous intellectual debt particularly relating to optimization and its application to system control to his instruction and guidance. Dr. Rathinam and Dr. Swaroop have shown great consideration and care not only as my academic advisors but also as personal mentors and companions, and I greatly appreciate everything they have done for me.

Special thanks to my committee members, Dr. Reza Langari, and Dr. Sergiy Butenko, for their support, guidance and helpful suggestions. Their guidance was invaluable in writing this dissertation and I owe them my heartfelt appreciation.

I was fortunate to have wonderful lab members, Kaarthik Sundar, Harsha Nagarajan, Manyam Satyanarayana, Navid Mohsenizadeh and Peyman Moghadas. Their passion for research, intellectual curiosity and attitude towards life have inspired me throughout the process.

I want to thank my parents, Sang Hoon Bae and In Soon Kwon, my sister Jin Yung Bae and brother-in-law Noah Popp for all their support. Their encouragement was critical in my decision to pursue a Ph.D in the United States. Their endless love enabled me to overcome any difficulties.

Finally, I thank Myoungkuk Park, my colleague and my husband. He is the one who gave me bottomless support for everything. I really appreciate that we could go through this tough time together. I have no doubt that, for the forthcoming journey of our lives, he will always be my best colleague and confidant in the world. I thank him for giving me the most precious baby, Jayna Park, the one that I live for and provide me with such happiness. I also want to thank to my parents in law for their endless prayers for us.

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGEMENTS iv

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES xi

1. INTRODUCTION 1
   - 1.1 Related Work 3
   - 1.2 Organization 6

2. APPROXIMATION ALGORITHMS FOR MULTIPLE TERMINAL HAMILTONIAN PATH PROBLEMS 7
   - 2.1 Multiple depot, multiple terminal, Hamiltonian path problems 7
     - 2.1.1 Problem statement 7
     - 2.1.2 Approximation algorithm for MDMTHPP 8
     - 2.1.3 Optimal constrained forest as a two matroid intersection problem 11
   - 2.2 Single depot multiple terminal Hamiltonian path problems 14
     - 2.2.1 Problem statement 14
     - 2.2.2 Approximation algorithm for SDMTHPP 15

3. AN APPROXIMATION ALGORITHM FOR A MULTIPLE DEPOT HETEROGENEOUS TRAVELING SALESMAN PROBLEM 23
   - 3.1 Problem statement 23
   - 3.2 Problem formulation 23
   - 3.3 Main ideas in the proposed approach 26
   - 3.4 Primal-Dual algorithm for MDHTSP 27
     - 3.4.1 Properties of the primal-dual algorithm 35
     - 3.4.2 Feasibility and running time analysis 37
     - 3.4.3 Approximation ratio analysis 38
   - 3.5 Simulation results 43

4. ALGORITHMS FOR A MULTIPLE DEPOT HETEROGENEOUS ASYMMETRIC TRAVELING SALESMAN PROBLEM 51
   - 4.1 Problem statement 51

4.2 Approximation algorithm using LP relaxation 52
4.2.1 Problem formulation 52
4.2.2 An approximation algorithm for the MDHATSP 54
4.3 A Primal-dual heuristic algorithm 59
4.3.1 Problem formulation 59
4.3.2 A Primal-dual heuristic algorithm 61
4.4 Computational results 70

5. CONCLUSIONS 74

REFERENCES 76

LIST OF FIGURES

FIGURE Page

2.1 Step 1 of $approx_1$: Find an optimal constrained forest. In this example, there are $m = 4$ depots and $n = 20$ targets. Also, $t_1 = 28$, $t_2 = 27$, $t_3 = 25$, $t_4 = 26$. 10

2.2 Step 2a of $approx_1$: For $i = 1, \cdots, 4$, double any edge that is not on the path from depot i to its terminal t_i in the tree T_i. 10

2.3 Step 2b of $approx_1$: For $i = 1, \cdots, 4$, find a walk and shortcut any target that is visited more than once such that the path starts from depot i, visits each of the targets in T_i exactly once before reaching the terminal t_i. 11

2.4 Step 1 of $approx_2$: Find an optimal constrained tree. In this example, there are $m = 4$ vehicles and $n = 24$ targets. 18

2.5 Step 2 of $approx_2$: The paths in $\mathcal{T}$ for each vehicle from the depot to their respective terminals. $PATH_i$ corresponds to the i^{th} vehicle at the depot. 19

2.6 Step 3 of $approx_2$: The graph G_T obtained by removing all the edges that lie on the paths, $PATH_1, \cdots, PATH_m$ from $\mathcal{T}$; in addition, any vertex whose degree is zero after the removal of the edges is also eliminated. 19

2.7 Odd degree targets of the graph G_T. 20

2.8 Step 4 of $approx_2$: The edges from the minimum cost, perfect matching on the odd-degree targets are added to G_T. The new graph is denoted by G_{new}. 20

2.9 Step 5 of $approx_2$: Add the edges of each connected component of G_{new} to some graph in $\{G_1, \cdots, G_m\}$ that shares a common target with the connected component. 21

2.10 Step 6 of $approx_2$: For $i = 1, \cdots, m$, find a walk that starts at the depot and ends at the terminal in G_i. Shortcut any previously visited targets so that each target is visited exactly once by some vehicle. . . 22

3.1 Snapshot of the forest for an example with 3 vehicles and 9 targets at the end of the first iteration of the main loop. The radius of the circular region, $p_i(u) := \sum_{S:u\in S} Y_i(S)$, around a target u in the forest F_i is equal to the sum of the dual variables of all the components that contain u in F_i. An edge between target 1 and 2 is added to F_1 as the corresponding constraint of (3.10) became tight. 31

3.2 Snapshot of the forest after nine iterations of the main loop. The constraint corresponding to the edge joining the target 8 and the depot 2 becomes tight. The edge is added to F_2 and the merged component is deactivated as the target 7 and 8 is now connected to the depot 2. The component $\{7, 8\}$ in F_3 is also deactivated. These components never become active again. 32

3.3 Snapshot of the forest after eleven iterations of the main loop. The component $\{7, 8\}$ in F_1 is deactivated and marked since the corresponding constraint in (3.11) became tight. 33

3.4 Snapshot of the forest after fifteen iterations of the main loop. The edge between the target 3 and 7 is added to F_1 as the corresponding constraint becomes tight. The component $\{7, 8\}$ in F_1 is now merged with the component $\{3\}$ and the new component $\{3, 7, 8\}$ in F_1 is now active again. 34

3.5 Snapshot of the forest at the end of the main loop 34

3.6 Snapshot of the final forest after the pruning step 35

3.7 Example instance with 2 vehicles 45

3.8 Example instance with 4 vehicles 46

3.9 Average approximation ratio for 50 instances 48

3.10 Maximum approximation ratio for 50 instances 49

3.11 Average running time for 50 instances 50

4.1 Snapshot of the forest for an example with 3 vehicles and 14 targets at the end of the first iteration of the main loop. At the beginning of the first iteration, the components which contain target 1 are chosen to increase the dual variables. An edge coming from target 7 to target 1 is added to F_1 as the corresponding constraint of (4.40) becomes tight. Since target 1 does not form any strongly connected component and is not reachable from d_1, $\{1\} \in \mathcal{C}_1$ is deactivated. 65

4.2 Snapshot of the forest after eight iterations of the main loop. The components that contain target 8 are chosen to increase the dual variables and an edge coming from target 5 to target 8 is added to F_1. Since target 5 and target 8 forms a strongly connected component, $\{5, 8\} \in \mathcal{C}_1$ becomes an active component. 65

4.3 Snapshot of the forest after eleven iterations of the main loop. The components that contain target 11 are chosen to increase the dual variables, and an edge coming from depot 2 to target 11 is added to F_2. Since the target 11 is now reachable from d_2, $\{11, d_2\} \in \mathcal{C}_2$ became an inactive component. The subset $\{11\} \in \mathcal{C}_3$ is deactivated and the superset $\{11\} \in \mathcal{C}_1$ is marked and deactivated. 66

4.4 Snapshot of the forest in the middle of the thirty ninth iteration of the main loop. In the first map, we can see that the components $\{13, 14\}, \{3\}, \{11\}$ are still violated components but inactive. 66

4.5 Snapshot of the forest after thirty nine iterations of the main loop. By following the combining procedure of the algorithm, now the component $\{4, 5, 8, 13, 14\} \in \mathcal{C}_1$ became an active violated component. . . 67

4.6 Snapshot of the forest in the middle of the forty first iteration of the main loop. The active component $\{4, 5, 8, 13, 14\} \in \mathcal{C}_1$ has all inactive subsets $\{4, 5, 8\}, \{13, 14\} \in \mathcal{C}_2$, but $\{13, 14\} \in \mathcal{C}_2$ is a violated component. 67

4.7 Snapshot of the forest after forty one iterations of the main loop. By following the combining procedure of the algorithm, now the component $\{4, 5, 8, 13, 14\} \in \mathcal{C}_2$ became an active violated component. . . . 68

4.8 Snapshot of the forest after the termination of the main loop. 68

4.9 Snapshot of the final forest after the pruning step 69

LIST OF TABLES

TABLE Page

4.1 Comparison of the theoretical and simulation results for $m = 2$ 71

4.2 Comparison of the theoretical and simulation results for $m = 3$ 72

4.3 Comparison of the theoretical and simulation results for $m = 4$ 72

4.4 Comparison of the computation time in seconds for $m = 2$ 73

4.5 Comparison of the computation time in seconds for $m = 3$ 73

4.6 Comparison of the computation time in seconds for $m = 4$ 73

1. INTRODUCTION*

Unmanned vehicles are commonly used in surveillance applications for monitoring and tracking a set of targets. For example, in the Cooperative Operations in Urban Terrain project [9] at the Air Force Research Laboratory, a team of unmanned vehicles are required to monitor a set of targets and send information/videos about the targets to the ground station. A human operator enters the locations of the targets through a human-machine interface, and the central computer associated with the interface has a few minutes to determine the motion plans for each of the vehicles. A fundamental subproblem that has to be solved by this computer is the problem of finding a tour for each vehicle so that each target is visited at least once by some vehicle and the sum of the distances traveled by all of the vehicles is minimal. This routing problem is known as the Traveling Salesman Problem (TSP) in the case where there is only one vehicle. In the case where there are multiple vehicles that possibly start from different initial locations or depots, this routing problem is known as the Multiple Depot TSP. Once the routing problem is solved and the tours have been determined, a nominal trajectory can be specified for each vehicle that includes other kinematic constraints of the vehicles, using the results in [23].

A multiple depot TSP is a generalization of the single TSP and is NP-hard[27]. The main focus of this dissertation is in developing fast algorithms that produce approximate solutions rather than finding optimal solutions, which may be relatively difficult. Therefore, the objective of this dissertation is on developing approximation algorithms for variants and generalizations of the multiple depot TSP. The routing

*Reprinted with permission from J. Bae and S. Rathinam, *Approximation algorithms for multiple terminal, Hamiltonian path problems*, Optimization Letters, vol. 6, no. 1, 69-85, January 2012, Copyright by Springer 2010.

problems considered in this dissertation are quite general and also arise in other surveillance applications such as crop monitoring [11], [15], and forest temperature monitoring [30].

Approximate solutions are found for the routing problems through the framework of approximation algorithms. An α-approximation algorithm is an algorithm that has a polynomial running time, and returns a solution whose cost is within α times the optimal cost for every instance of the problem. The factor α is generally referred as the approximation ratio.

The dissertation addresses the following three variants and generalizations of the multiple depot TSP:

- A symmetric, multiple depot, multiple terminal, Hamiltonian path problem (MDMTHPP) is considered in section 2 and can be stated as follows: Given m vehicles that start from distinct depots, m terminals and n targets, the problem is to choose paths for each of the vehicles so that (1) each vehicle starts at his respective depot (or initial location), visits at least one target and reaches any one of the terminals not visited by other vehicles, (2) each target is visited exactly once and (3) the sum of the distances travelled by the vehicles is minimized. We propose an approximation algorithm for the MDMTHPP if the costs are symmetric and satisfy the triangle inequality. First, an algorithm that runs in $O((n+2m)^3)$ steps with approximation ratio of 2 is presented. Next, we consider a special case of the problem where all the vehicles start from a single depot, known as the single depot, multiple terminal, Hamiltonian path problem (SDMTHPP). For the SDMTHPP, an algorithm that runs in $O((n+2m)^3)$ steps with approximation ratio of $\frac{5}{3}$ is proposed.

- A multiple depot, heterogeneous traveling salesman problem (MDHTSP) is considered in section 3 and can be stated as follows: Given m heterogeneous vehicles that start from distinct depot and n targets, the problem is to choose a tour for each vehicle such that each target is visited at least once by a vehicle and the sum of the distances travelled by the vehicles is minimized. Determining a tour for a vehicle specifies a sequence of targets to visit for the vehicle. Vehicles are considered to be heterogeneous if the distance to travel between any two targets depends on the type of the vehicle used. We propose a 2-approximation algorithm for an important special case of this MDHTSP when the vehicles considered as ground vehicles that can move forwards and backwards with a minimum turning radius.

- A multiple depot, heterogeneous asymmetric salesman problem (MDHATSP) is considered in section 4 and can be stated as follows: Given a team of small heterogeneous vehicles located at distinct depots, a set of target locations and the cost of traveling between any two locations for each UV, find a tour for each vehicle such that each target is visited by a vehicle and the sum of the travel cost of all the vehicles is minimized. First, for the MDHATSP with m vehicles and n targets, we present a $mlog_2(n+1)$-approximation algorithm when the travel costs are asymmetric and satisfy the triangle inequality. Later, we develop a primal-dual heuristic for the problem for a special case when the vehicles considered are fixed-wing Unmanned Aerial Vehicles (UAVs). Simulation results are also presented to corroborate the performance of the proposed algorithms.

1.1 Related Work

The MDMTHPP is a generalization of the single TSP. TSP's have received significant attention in the area of combinatorial optimization [18],[27]. In general, it is known that there cannot exist a constant factor approximation algorithm for a TSP unless $P = NP$ [27]. However, when the costs satisfy the triangle inequality, there are constant factor approximation algorithms available for the single TSP and the Hamiltonian Path Problems (HPPs). The algorithm by Christofides [3] with an approximation factor of $\frac{3}{2}$ is currently one of the best known approximation algorithms for the single TSP when the costs satisfy the triangle inequality. This approximation algorithm has been extended by Hoogeveen to variants of the single HPP in [14]. Specifically, Hoogeveen developed a $\frac{5}{3}$-approximation algorithm for a Single Depot, Single Terminal Hamiltonian Path Problem in [14].

In general, there are two basic combinatorial problems while dealing with a multiple salesman problem. One is a partitioning problem that requires a subset of targets to be assigned for each salesman to visit. The second is the sequencing problem of finding a suitable sequence of visiting the subset of targets assigned to each salesman. The difficulty in dealing with a multiple salesman problem is that both the partitioning problem and the sequencing problem are coupled. Currently, there are 2-approximation algorithms for variants of the multiple salesman problem available in the literature [17], [23], [21]. Specifically, Rathinam et al. present a 2-approximation algorithm for the MDMTHPP that runs in $O((n + 2m)^6)$ steps in [21]. Each of the 2-approximation algorithms works in a sequence of two main steps. In the first step, a constrained forest problem which is generally a relaxation of the given multiple salesman problem is solved optimally. In the second step, edges in the constrained forest are doubled to obtain an Eulerian graph for each salesman. A path or a tour can then be suitably found for each salesman using the Eulerian graphs by shortcutting any target that is visited more than once. The computational complexity of a 2-approximation algorithm crucially depends on how fast one can find the optimal constrained forest. In section 2, we pose this constrained forest problem corresponding to the MDMTHPP as a weighted, two-matroid intersection problem where one of the matroids in a partition matroid. Therefore, one can use the specialized algorithm by Brezovec et al. [1] to find the optimal constrained forest in $O((n+2m)^3)$ steps.

The generalization of the Christofides algorithm that can yield an approximation factor of $\frac{3}{2}$ to the case where there are multiple salesman and the salesmen start from distinct depots and end at distinct terminals is currently an open problem. However, there are some cases for which the extension of the Christofides algorithm is available. In [8], Frieze presents a $\frac{3}{2}$-approximation algorithm for the case when all the depots and the terminals are at the same location and the objective is to find a tour for each salesman such that each salesman visits at least one target, each target is visited by some salesman and the total cost of the tours is a minimum. In [22], Rathinam et al. present a $\frac{3}{2}$-approximation algorithm for variants of a 2 depot, HPP. When the number of depots is a constant and each salesman is required to return to his starting depot, Xu et al. [26] have developed a novel $\frac{3}{2}$-approximation algorithm for the multiple depot, TSP. In section 2, we develop a $\frac{5}{3}$-approximation algorithm for the SDMTHPP where all the vehicles start from the same depot but can terminate their paths at distinct terminals.

When the vehicles are heterogeneous, there are currently no constant factor approximation algorithms in the literature even if the costs are symmetric and satisfy the triangle inequality. Currently, there are algorithms whose approximation factors scale linearly in the number of heterogeneous vehicles [4],[29]. In section 3, we present a 2-approximation algorithm for the case when the travel costs are symmetric, satisfy the triangle inequality and a monotonicity property. In section 4, for a problem with m vehicles and n targets, we present a $m\log_2(n+1)$-approximation algorithm for the case when the travel costs are asymmetric and satisfy the triangle inequality. This result generalizes the $\log_2(n)$-approximation algorithm for the single vehicle problem in [7].

1.2 Organization

Each of the sections (2,3,4) formally introduces the problem, presents the approximation algorithm and proves the corresponding approximation ratio of the proposed algorithms. The final section concludes the dissertation with directions for future work.

2. APPROXIMATION ALGORITHMS FOR MULTIPLE TERMINAL HAMILTONIAN PATH PROBLEMS*

2.1 Multiple depot, multiple terminal, Hamiltonian path problems

2.1.1 Problem statement

Let m vehicles be initially located at depots represented by $D := \{1, 2, \cdots, m\}$. Let $U := \{m+1, m+2, \cdots, m+n\}$ be vertices representing the targets. Assume $n \leq m$. Let $P := \{m+n+1, m+n+2, \cdots, 2m+n\}$ be the set of vertices of possible terminals. Let $V = D \cup U \cup P$. E denotes the set of all the undirected edges joining any two vertices in V. Let $\{i, j\}$ represent the undirected edge joining vertices i and j and $C(\{i, j\})$ denote the cost of traveling $\{i, j\}$. Let $C_{max} = \max_{i,j \in V} C(\{i, j\})$. Costs are assumed to be positive and satisfy the triangle inequality. A path for the i-th vehicle is denoted by an ordered sequence of vertices, $PATH_i = \{d_i, v_{i1}, v_{i2}, \cdots, v_{ik_i}, t_i\}$, where $d_i \in D$ is the depot associated with the vehicle, $t_i \in P$ is the terminal visited by the vehicle, k_i is the number of targets visited by the vehicle, and $v_{ij} \in U$ is the target visited at j-th order by the vehicle for all $j \in \{1, \cdots, k_i\}$. The i-th vehicle starts from depot d_i, visits target v_{i1}, then v_{i2}, and so on until it reaches the terminal t_i. The cost of the path $PATH_i$ is defined as $C(PATH_i) = C(\{d_i, v_{i1}\}) + \sum_{j=1, j\geq 1}^{k_i - 1} C(\{v_{ij}, v_{i(j+1)}\}) + C(\{v_{ik_i}, t_i\})$. The problem needs to find a path for each vehicle such that each target is visited by some vehicle, each vehicle visits at least one target before reaching a terminal not visited by any other vehicle while the sum of all the costs traveled by the vehicles is minimized. A feasible solution of the MDMTHPP will have the following properties:

*Reprinted with permission from J. Bae and S. Rathinam, *Approximation algorithms for multiple terminal, Hamiltonian path problems*, Optimization Letters, vol. 6, no. 1, 69-85, January 2012, Copyright by Springer 2010.

- There will be exactly m paths, $PATH_1, \cdots, PATH_m$, in the solution where the degree of each of the depots and terminals is equal to one.
- Each path visits at least one target.
- Each target is visited exactly once and the degree of each of the targets in the solution is equal to two.

2.1.2 Approximation algorithm for MDMTHPP

Since the crux of the algorithm is the computation of an optimal constrained forest, we first define it before presenting the approximation algorithm for MDMTHPP. A constrained forest associated with the given set of vertices $V = D \cup U \cup P$ is a collection of trees such that

- there is no path connecting any two depots or any two terminals,
- there are exactly m trees with at least one target in each tree,
- the degree of each of the depots and the terminals is exactly one.

An *optimal* constrained forest is a constrained forest where the sum of the cost of all the edges in the forest is a minimum.

The approximation algorithm, $approx_1$, for the MDMTHPP is presented as follows in **Algorithm 1**. In Figure 2.1, 2.2, and 2.3, we can see how $approx_1$ is processed with an example.

Clearly, $approx_1$ produces a feasible solution for the MDMTHPP. Since the cost of the edges satisfy the triangle inequality, shortcutting procedure does not increase the cost of the paths. Therefore, the total cost of the Hamiltonian Paths that $approx_1$ provide must be upper bounded by twice of the cost of the optimal constrained forest.

Algorithm 1 *$approx_1$*

```
1: Find an optimal constrained forest. Let $T_i$ be the tree corresponding to the
   vehicle at the $i$th depot in this forest. Let $t_i$ denote the terminal present in the
   tree $T_i$. This step essentially solves the partitioning problem; the vehicle at the
   $i$-th depot must visit each of the targets in $T_i$ before reaching $t_i$.
2: for $i = 1, \cdots, m$ do
3:     (a) Double any edge in $T_i$ that is not on the path from $i$ to $t_i$. Note that in
       this new graph, denoted by $\mathcal{E}_i$, all the vertices have an even degree except the
       depot $d_i$, and the terminal $t_i$.
4:     (b) Using the graph $\mathcal{E}_i$, find a Eulerian walk that starts from $d_i$, visits each
       of the edges in $\mathcal{E}_i$ exactly once before reaching the terminal $t_i$. Shortcut this
       walk to find a Hamiltonian path that starts from $d_i$, visits each of the targets
       in $T_i$ exactly once before reaching the terminal $t_i$.
5: end for
```

As finding an optimal constrained forest is a relaxation of the MDMTHPP by dropping the degree constraints on the targets, the total cost of the Hamiltonian Paths found by *$approx_1$* must be at most equal to twice the optimal cost of the MDMTHPP. It is also clear that the computational complexity of *$approx_1$* is dominated by the first step of *$approx_1$* which requires one to compute an optimal constrained forest. This part of the proof is discussed in the next subsection.

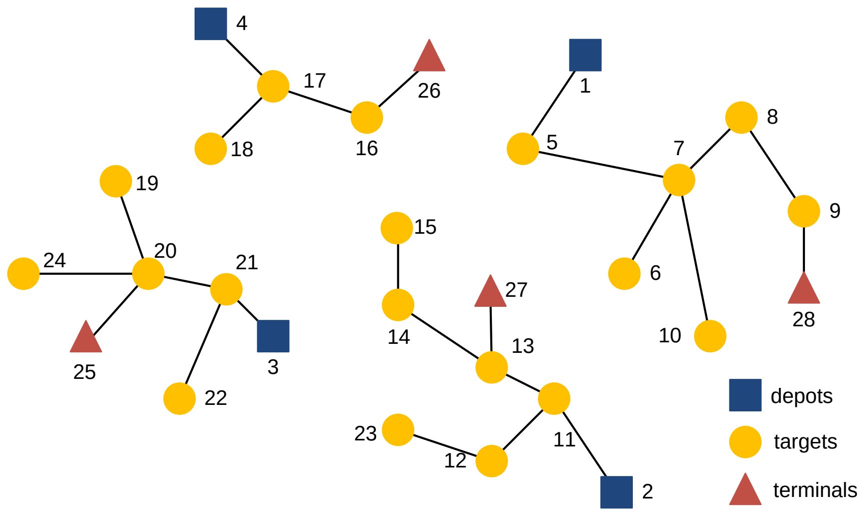

Figure 2.1: Step 1 of $approx_1$: Find an optimal constrained forest. In this example, there are $m = 4$ depots and $n = 20$ targets. Also, $t_1 = 28$, $t_2 = 27$, $t_3 = 25$, $t_4 = 26$.

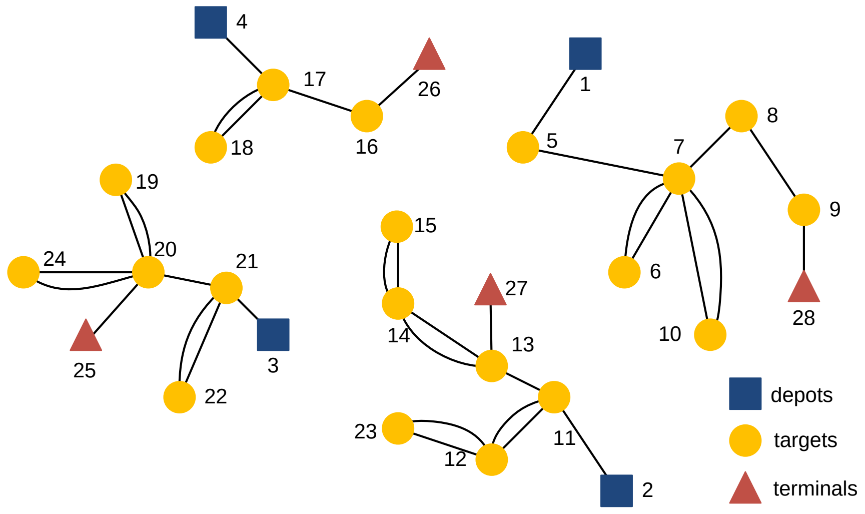

Figure 2.2: Step 2a of $approx_1$: For $i = 1, \cdots, 4$, double any edge that is not on the path from depot i to its terminal t_i in the tree T_i.

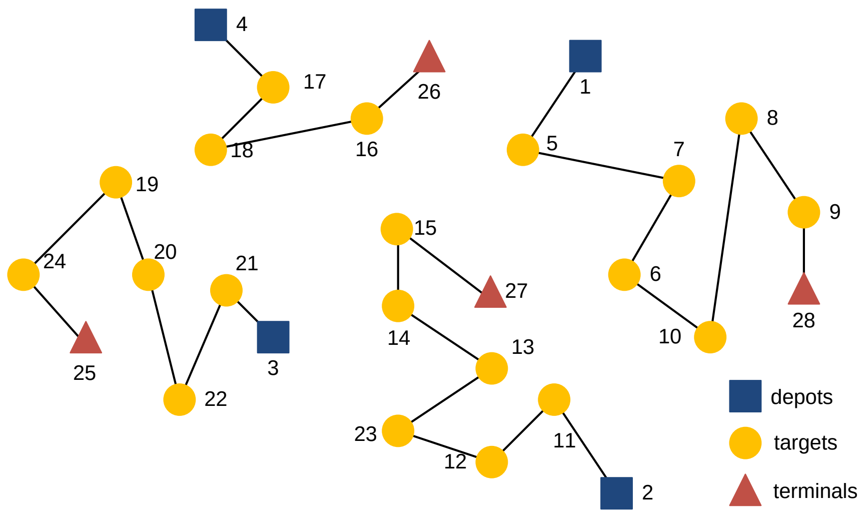

Figure 2.3: Step 2b of $approx_1$: For $i = 1, \cdots, 4$, find a walk and shortcut any target that is visited more than once such that the path starts from depot i, visits each of the targets in T_i exactly once before reaching the terminal t_i.

2.1.3 Optimal constrained forest as a two matroid intersection problem

The basic idea is to formulate the problem of finding an optimal forest in $\mathbb{G} = (V, E)$ as equivalent to finding a directed forest in another graph $\mathbb{G}'$. Then, it is shown that the problem of finding a minimum cost, directed forest in $\mathbb{G}'$ can be posed as a weighted, two matroid intersection problem where one of the matroids is a partition matroid. Hence, one can use the specialized algorithm by Brezovec et al. [1] to solve the matroid intersection problem, and as a result, obtain an optimal constrained forest.

$\mathbb{G}' = (V', E')$ is a directed graph where $V' := D \bigcup U \bigcup P$, and E' consists of a set of directed edges joining the vertices in V' and is defined as follows.

$$E' := \{(i,j),\ \forall\ i \in D \text{ and } \forall j \in U\} \bigcup \{(i,j),\ \forall\ i,j \in U, i \neq j\} \bigcup \{(i,j),\ \forall i \in U \text{ and } j \in P\}. \tag{2.1}$$

In this directed graph $\mathbb{G}'$, (i,j) denotes a directed edge from vertex i to vertex j. Note that in $\mathbb{G}'$, there is no edge directed into any of the depots and there is no outgoing edge from any of the terminals. Also, the cost of edges in $\mathbb{G}'$ are defined as follows:

$$C'(i,j) = \begin{cases} C(\{i,j\}), \forall i \in D, \forall j \in U, \\ C(\{i,j\}), \forall i,j \in U, i \neq j, \\ C(\{i,j\}), \forall i \in U, \forall j \in P. \end{cases} \tag{2.2}$$

Now define a directed forest $\mathcal{F} = (V', B)$ where $B \subseteq E'$ as follows:

- $\mathcal{F}$ consists of m trees with exactly one depot, one terminal and at least one target in each tree.
- The in-degree of each of the vertices in $\mathcal{F}$ is at most equal to one.
- The out-degree of each of the depots and the in-degree of each of the terminals is exactly equal to one.

Note that there is a one to one correspondence between the set of all the constrained forests in $\mathbb{G}$ and the set of all the directed forests in $\mathbb{G}'$. Therefore, finding an optimal constrained forest in $\mathbb{G}$ is equivalent to finding a minimum cost, directed forest in $\mathbb{G}'$. Now, we will pose the problem of finding the minimum cost, directed forest as a problem of finding a common base in the intersection of two matroids.

Let S_i be the set of all the edges directed into $i \in V'$. Let $\mathcal{I'}_1$ be a collection of subsets of E' such that the number of edges in each subset directed into i is at most equal to 1. That is, $\mathcal{I'}_1 := \{A_1 : A_1 \subseteq E', |A_1 \bigcap S_i| \leq 1 \ \forall i \in V'\}$. It is well known that $M'_1 = (E', \mathcal{I'}_1)$ is a partition matroid [18].

Now, for any subset $B \subseteq E'$, let B_{ud} represent the undirected counterpart of B obtained by disregarding the directions of the edges in B. Essentially, if there are two directed edges $e_1 = (i, j)$ and $e_2 = (j, i)$ in B, then we get two undirected edges also labeled e_1 and e_2 between i and j in B_{ud}. Let $\mathcal{I'}_2$ be a collection of subsets of E' such that $E' \supseteq B \in \mathcal{I'}_2$ if and only if graph (V', B_{ud}) is free of cycles and free of paths connecting any pair of terminals in P. It is known that $M'_2 = (E', \mathcal{I'}_2)$ is a matroid (Cerdeira [2], Waqar et al. [19]).

Lemma 2.1. *Consider any common base, $\mathfrak{B}$, in the intersection of matroids M'_1 and M'_2. Each tree in this base will consist of at most one depot.*

Proof. We will prove this lemma by contradiction. Suppose there is a tree $\mathcal{T}$ in $\mathfrak{B}$ that consists of at least two depots. Let two of these depots in $\mathcal{T}$ be denoted by d_1 and d_2. Also, let the terminal vertex present in $\mathcal{T}$ be denoted by t. Since there is no incoming edge into d_1 and there is no outgoing edge from t, and since the in-degree of each of the vertices in $\mathcal{T}$ must be at most equal to 1, there must be a directed path from depot d_1 to terminal t. By a similar argument, one can also deduce that there must be a directed path from depot d_2 to terminal t. But this is not possible again due to the fact that the in-degree of each vertex in $\mathcal{T}$ must be at most equal to one. Therefore, any tree in $\mathfrak{B}$ cannot have more than one depot. Hence proved. $\square$

As there are exactly m trees and m depots in any common base, it follows from the above lemma that each tree in any common base will consist of exactly one depot.

Therefore, any common base in the intersection of matroids M'_1 and M'_2 will have the following properties:

- The base will consist of m trees with a depot and a terminal in each tree.
- As there is no edge joining a depot and a terminal in E', each tree must consist of at least one target.

Now, it is clear that this common base satisfies all the requirements in a directed forest except for the degree constraints that state that the number of edges incident on each of the depots and the terminals must be equal to one. To meet this requirement, add a large constant say $\lambda := (n + 2m)C_{max}$ to the cost of each of the edges incident on all the depots and the terminals. Now, any minimum cost base in the intersection of matroids M'_1 and M'_2 with the modified cost will not have more than one edge incident on any of the depots and the terminals. Therefore, finding an optimal directed forest can be posed as finding an optimal base in the intersection of matroids M'_1 and M'_2 with the modified cost. Since M'_1 is a partition matroid, one can use the specialized algorithm by Brezovec et al. [1] to find the optimal base. This algorithm runs in $O((n + 2m)^3)$ steps.

We now summarize one of the main results of this section in the following theorem:

Theorem 2.1. *The algorithm* $\boldsymbol{approx}_1$ *solves the MDMTHPP with an approximation factor of 2. Moreover, the number of steps required is of* $O((n + 2m)^3)$ *where* n *is the number of targets and* m *is the number of depots.*

2.2 Single depot multiple terminal Hamiltonian path problems

2.2.1 Problem statement

Let m vehicle be initially located at the same depot represented by $D := \{1\}$. Let $U := \{2, 3, \cdots, n + 1\}$ be vertices representing the targets. Assume $n \leq m$. Let

$P := \{n+2, n+3, \cdots, m+n+1\}$ be the set of vertices of possible terminals. Let $V = D \cup U \cup P$. All the other notations used for the SDMTHPP are same with the ones used for the MDMTHPP. The objective of the problem is to find a path for each vehicle such that each target is visited by some vehicle, each vehicle visits at least one target before reaching a terminal not visited by any other vehicle, and the sum of the costs of the paths traveled by the vehicles is minimized.

2.2.2 Approximation algorithm for SDMTHPP

The approximation algorithm, $approx_2$, for the SDMTHPP is as follows in **Algorithm 2**. Figure 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, and 2.10 show how $approx_2$ works at each step with an example.

It is clear that each of the paths obtained using the above algorithm starts at the depot, visits at least one target and finally reaches a distinct terminal. Also, each of the targets is visited exactly once. Hence, the above algorithm, $approx_2$, finds a feasible solution, sol_f, for the SDMTHPP. The computational complexity of $approx_2$ is dominated by the number of steps required to find the optimal constrained tree and the perfect matching. Using the algorithm presented in the previous section, the optimal constrained tree can be found in $O(n+2m)^3$ steps. The minimum cost, perfect matching found by Edmonds algorithm requires $O(n^3)$ steps [6][10]. Therefore, the number of steps required to implement $approx_2$ is of $O((n+2m)^3)$.

The remaining part of this section aims to prove that $Cost(sol_f) \leq \frac{5}{3} C^*_{opt}$ where $Cost(sol_f)$ is the sum of the cost of the edges in sol_f and C^*_{opt} is the optimal cost of the SDMTHPP. This would prove that the approximation ratio of $approx_2$ is $\frac{5}{3}$. Now, since the costs satisfy the triangle inequality, short cutting the targets in the last step of $approx_2$ will not increase the cost of the paths. Therefore, $Cost(sol_f)$, obtained using $approx_2$ is upper bounded by $Cost(M_{opt}) + Cost(\mathcal{T})$ where $Cost(M_{opt})$

Algorithm 2 *$approx_2$*

```
1: Find an optimal constrained tree, $\mathcal{T}$, covering the depot, all the targets and the
   terminals such that
   -the degree of the depot is $m$,
   -the path joining any two terminals passes through the depot,
   -the depot is not adjacent to any of the terminals, and
   -the sum of the cost of the edges in the tree is a minimum.
2: Find the path joining the depot to each of the terminals in $\mathcal{T}$. These paths are
   denoted by $PATH_1, \cdots, PATH_m$. Each path will start at the depot, visit at
   least one target and finally reach a terminal. For $i = 1, \cdots, m$, we define the
   graph $G_i$ corresponding to $PATH_i$ as follows: A vertex is present in $G_i$ if and
   only if the vertex belongs to $PATH_i$, and an edge is present in $G_i$ if and only if
   the edge joins two adjacent vertices in $PATH_i$.
3: Obtain a graph $G_T$ from $\mathcal{T}$ by applying the following sequence of the operations:
   remove all the edges that lie on each of the paths, $PATH_1, \cdots, PATH_m$ from
   $\mathcal{T}$; remove all the vertices from $\mathcal{T}$ that have no edges incident on them. Observe
   that $G_T$ will consist of only target vertices and the parity of the degree of any
   target vertex $u$ in $G_T$ will be the same as the parity of $u$ in $\mathcal{T}$.
4: Let the set of all the odd-degree targets in the graph $G_T$ be denoted by $O$. Note
   that $|O|$ is even. Find a minimum cost, perfect matching, $\mathcal{M}_{opt}$, on all the targets
   in $O$ using Edmond's algorithm. Add the edges in $\mathcal{M}_{opt}$ to the edges in $G_T$ to
   form a new graph $G_{new}$. At this point, all the target of $G_{new}$ must have even
   degree. Let the connected components in $G_{new}$ be denoted by $C_1, C_2, \cdots, C_r$.
5: for $j = 1, \cdots, r$ do
6:     Let $G_{k_j}$ be a graph in $\{G_1, \cdots, G_m\}$ that shares a common target with the
       connected component $C_j$. Then do $G_{k_j} := G_{k_j} + C_j$ while the operation $H_1 + H_2$
       is defined as $H_1 + H_2 = (V_1 \cup V_2, E_1 \cup E_2)$, where $H_1 = (V_1, E_1)$ and $H_2 =
       (V_2, E_2)$ are the disjoint union of graphs. Since $C_j$ is an Eulerian graph, all
       the vertices in $G_{k_j}$ will have even degree except the depot and the terminal
       present in $G_{k_j}$.
7: end for
```

is the sum of the cost of the edges in the optimal matching M_{opt}, and $Cost(\mathcal{T})$ is the sum of the cost of the edges in the optimal, constrained tree $\mathcal{T}$. Also, observe that $Cost(\mathcal{T}) \leq C^*_{opt}$ since the problem of finding an optimal, constrained tree is a relaxation of the SDMTHPP without the degree constraints on the target vertices. Therefore, to show that $Cost(sol_f) \leq \frac{5}{3}C^*_{opt}$, the only remaining part is to show that $Cost(M_{opt}) \leq \frac{2}{3}C^*_{opt}$.

Bound on the cost of matching

As defined before, G_T is the forest obtained from $\mathcal{T}$ after the following sequence of operations: 1) the removal of all the edges that lie on each of the paths, $PATH_1$,$PATH_2$, $\cdots$,$PATH_m$, from $\mathcal{T}$; 2) the removal of all the zero-degree vertices. If $Cost(G_T)$ denotes the sum of the cost of the edges in G_T, using the results in [24], it is known that

$$Cost(G_T) \geq Cost(M_{opt}).$$

The set of edges that lie on the paths, $PATH_1, PATH_2, \cdots, PATH_m$, can be added to the optimal solution of the SDMTHPP to form an Eulerian graph that spans the depot, all the destinations and the terminals. Therefore, due to Euler's theorem, one can find an Eulerian walk and then find a tour spanning *only* the odd-degree destinations in O by short cutting any unnecessary vertex in the walk. As in the proof of the Christofides algorithm [3], this tour spanning the odd-degree destinations can further be decomposed into two disjoint sets of perfect matchings on the odd-degree destinations. Again, since the costs satisfy the triangle inequality, we have

$$\sum_{i=1}^{m} Cost(PATH_i) + C^*_{opt} \geq 2Cost(M_{opt}).$$

Hence, from the above arguments, it follows that

$$\begin{aligned} 2C^*_{opt} &\geq C^*_{opt} + Cost(\mathcal{T}) \\ &= C^*_{opt} + \sum_{i=1}^{m} Cost(PATH_i) + Cost(G_T) \\ &\geq 2Cost(M_{opt}) + Cost(M_{opt}) \\ &= 3Cost(M_{opt}). \end{aligned} \tag{2.3}$$

We now summarize the main result of this section:

Theorem 2.2. *The algorithm* ***approx***$_2$ *solves the SDMTHPP when the vehicles start from the same depot with an approximation factor of* $\frac{5}{3}$*. Moreover, the number of steps required to implement* ***approx***$_2$*, is of* $O((n+2m)^3)$ *where* n *is the number of targets and* m *is the number of depots.*

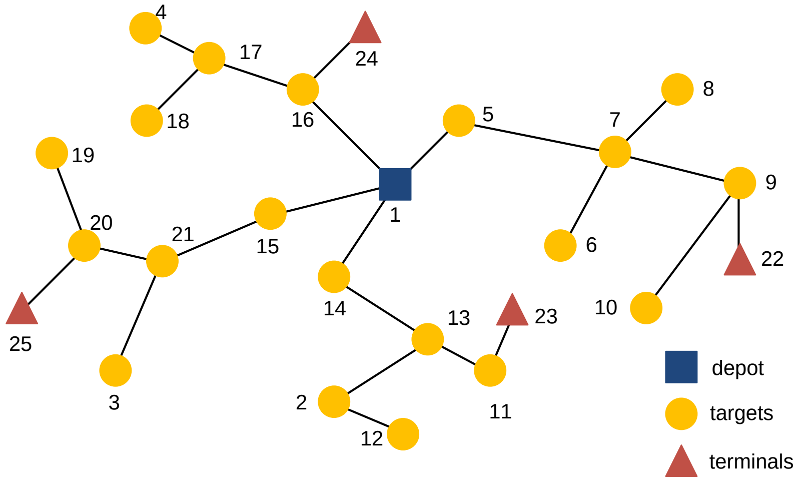

Figure 2.4: Step 1 of *approx*$_2$: Find an optimal constrained tree. In this example, there are $m = 4$ vehicles and $n = 24$ targets.

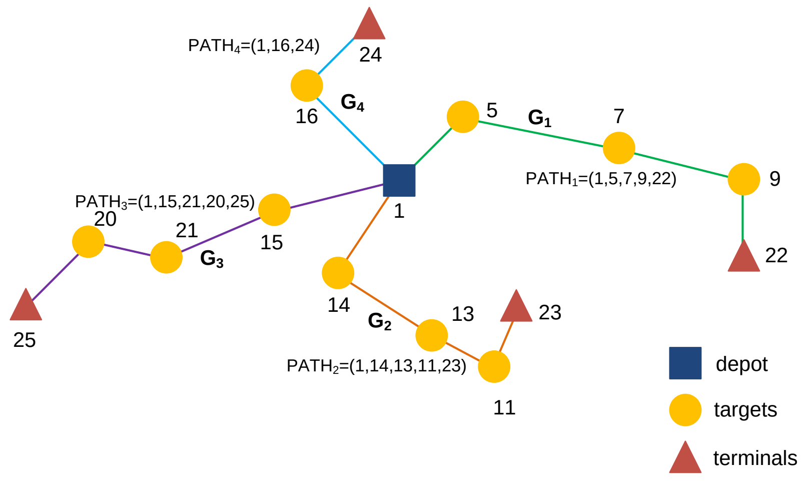

Figure 2.5: Step 2 of *approx*$_2$: The paths in $\mathcal{T}$ for each vehicle from the depot to their respective terminals. $PATH_i$ corresponds to the i^{th} vehicle at the depot.

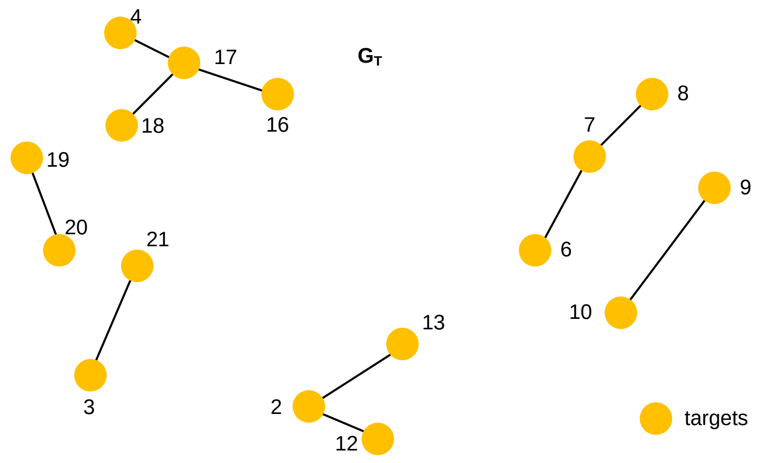

Figure 2.6: Step 3 of *approx*$_2$: The graph G_T obtained by removing all the edges that lie on the paths, $PATH_1, \cdots, PATH_m$ from $\mathcal{T}$; in addition, any vertex whose degree is zero after the removal of the edges is also eliminated.

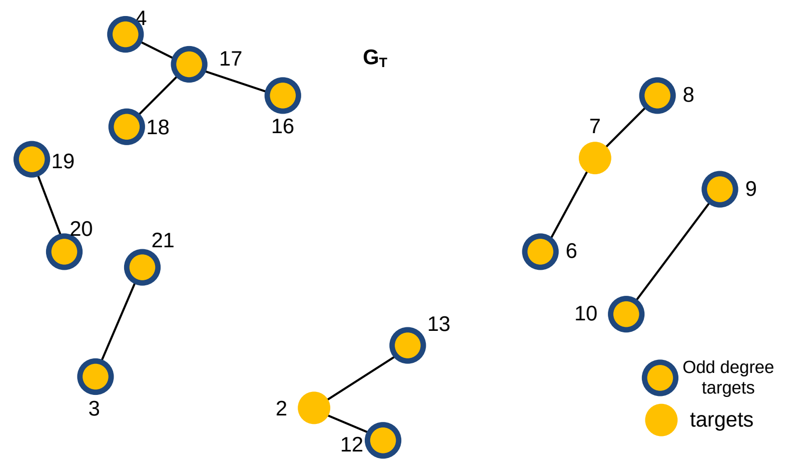

Figure 2.7: Odd degree targets of the graph G_T.

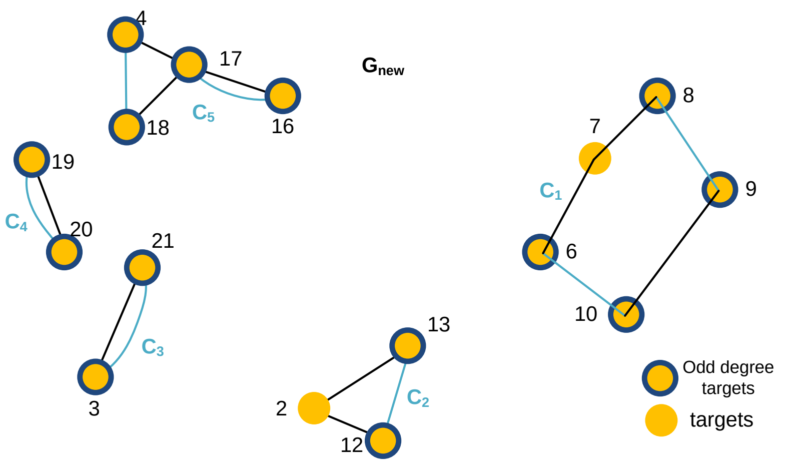

Figure 2.8: Step 4 of $approx_2$: The edges from the minimum cost, perfect matching on the odd-degree targets are added to G_T. The new graph is denoted by G_{new}.

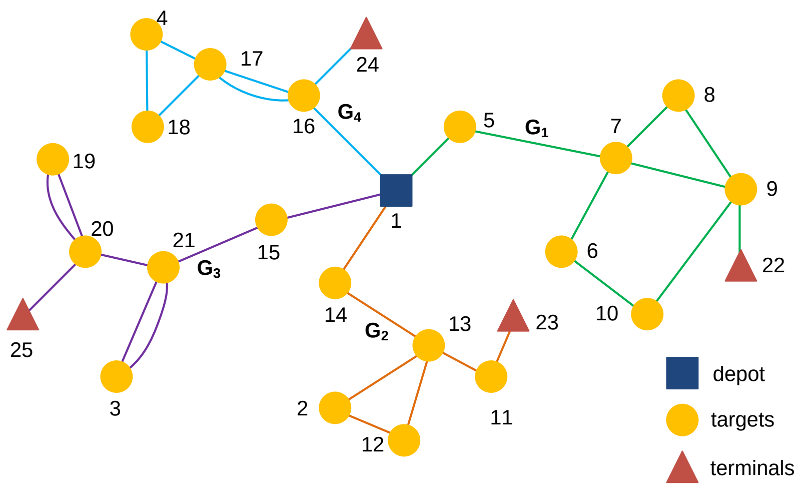

Figure 2.9: Step 5 of $approx_2$: Add the edges of each connected component of G_{new} to some graph in $\{G_1, \cdots, G_m\}$ that shares a common target with the connected component.

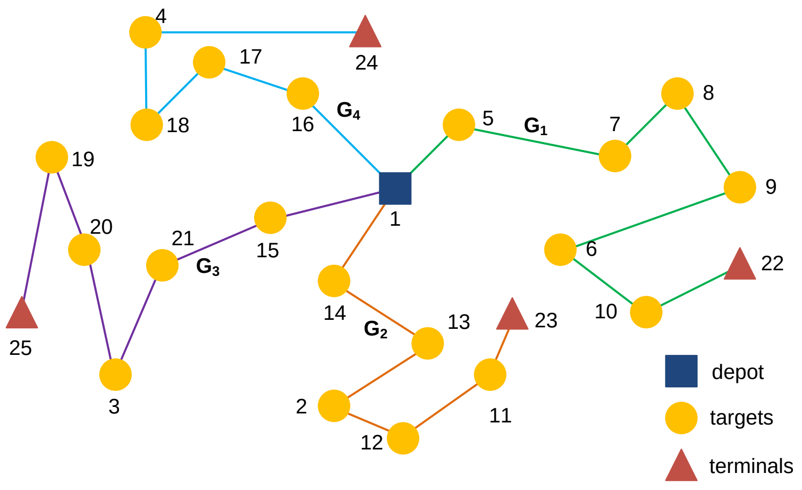

Figure 2.10: Step 6 of $approx_2$: For $i = 1, \cdots, m$, find a walk that starts at the depot and ends at the terminal in G_i. Shortcut any previously visited targets so that each target is visited exactly once by some vehicle.

3. AN APPROXIMATION ALGORITHM FOR A MULTIPLE DEPOT HETEROGENEOUS TRAVELING SALESMAN PROBLEM

3.1 Problem statement

Let the number of vehicles be m and let $D = \{d_1, d_2, \cdots, d_m\}$ be the set of vertices that correspond to the initial depots of the vehicles. Let $T = \{1, \cdots, n\}$ be the set of vertices that denotes all the targets. For each $k \in \{1, \cdots, m\}$, let $V_k := \{d_k\} \cup T$ denote the set of all the vertices corresponding to the k^{th} vehicle, and let E_k be the set of all the edges joining any two vertices in V_k. The cost to travel an edge $e \in E_k$ is represented by $cost_e^k$. All the costs are assumed to be positive and satisfy the triangle inequality. We also assume that the travel costs satisfy the following monotonicity property: For the edge e joining any two targets, $cost_e^1 \leq cost_e^2 \leq \cdots \leq cost_e^m$. A tour for a vehicle starts from its depot, visits a set of targets in a sequence and finally returns to its initial depot. The objective of the MDHTSP is to find a path for each vehicle such that each target is visited exactly once by some vehicle and the sum of the travel costs of all the vehicles is minimized.

3.2 Problem formulation

Let x_e^k be an integer variable that represents whether edge $e \in E_k$ is present in the tour corresponding to the k^{th} vehicle. For any edge e joining *two targets*, x_e^k can take values only in the set $\{0, 1\}$; $x_e^k = 1$ if e is present in the tour of the k^{th} vehicle, otherwise $x_e^k = 0$. In order for a tour to visit just one target if required, x_e^k is allowed to choose any of the values in the set $\{0, 1, 2\}$ for an edge e joining the depot d_k and a target $v \in T$. For $k = 1, \cdots, m-1$, let z_U^k denote a binary variable that determines the partition of the targets connected to each depot; z_U^k is equal to 1 if U contains all the targets that are not visited by the vehicles $\{1, \cdots, k\}$ and is equal to 0, otherwise. Let $\delta_k(S)$ for $k = 1, \cdots, m$, represent the subset of all the edges of E_k with one end in S and another end in $V_k \backslash S$. $\delta_k(S)$ is also refereed to as the cut set of corresponding to the k^{th} vehicle.

Consider the first vehicle. For any $S \subseteq T$, at least two edges must be chosen from $\delta_1(S)$ for the tour of the first vehicle if there is at least one vertex in S, that is not connected to any of the depots in the set $\{d_2, \cdots, d_m\}$, i.e., $\sum_{e \in \delta_1(S)} x_e^1 \geq 2$ if $\sum_{T \supseteq U \supseteq S} z_U^1 = 0$. This requirement can be written as $\sum_{e \in \delta_1(S)} x_e^1 \geq 2 - 2\sum_{T \supseteq U \supseteq S} z_U^1$. Similarly, for any vehicle k in the set $\{2, \cdots, m\}$, for any $S \subseteq T$, at least two edges must be chosen from $\delta_k(S)$ for the tour of the k^{th} vehicle if S contains at least one target that is visited by the k^{th} vehicle. This constraint can be written as $\sum_{e \in \delta_k(S)} x_e^k \geq 2\sum_{T \supseteq U \supseteq S}(z_U^{k-1} - z_U^k)$ for vehicles $k \in \{2, \cdots, m-1\}$, and $\sum_{e \in \delta_m(S)} x_e^m \geq 2\sum_{T \supseteq U \supseteq S} z_U^{n-1}$ for the m^{th} vehicle. Now, the MDHTSP without the degree constraints can be formulated as an integer linear program as follows:

$$\min \sum_{k=1,\cdots,m} \sum_{e \in E_k} cost_e^k \; x_e^k \tag{3.1}$$

$$\sum_{e \in \delta_1(S)} x_e^1 \geq 2 - 2 \sum_{T \supseteq U \supseteq S} z_U^1 \qquad \forall S \subseteq T, \tag{3.2}$$

$$\sum_{e \in \delta_k(S)} x_e^k \geq 2 \sum_{T \supseteq U \supseteq S} (z_U^{k-1} - z_U^k) \quad \forall S \subseteq T, \; and \; k = 2, \cdots, m-1, \tag{3.3}$$

$$\sum_{e \in \delta_m(S)} x_e^m \geq 2 \sum_{T \supseteq U \supseteq S} z_U^{m-1} \qquad \forall S \subseteq T, \tag{3.4}$$

$$x_e^k \in \{0, 1\} \text{ for } e = \{u, v\}_k \; \forall u, v \in T, \; where \; k = 1, \cdots, m$$

$$x_e^k \in \{0, 1, 2\} \text{ for } e = \{d_k, v\}_k \; \forall v \in T, \; where \; k = 1, \cdots, m$$

$$z_U^k \in \{0, 1\} \; \forall U \subseteq T, \; and \; k = 1, \cdots, m-1.$$

Consider a Linear Programming (LP) relaxation of the above problem where the integer constraints are relaxed.

$$\min \sum_{k=1,\cdots,m} \sum_{e \in E_k} cost_e^k \; x_e^k \tag{3.5}$$

$$\sum_{e \in \delta_1(S)} x_e^1 \geq 2 - 2 \sum_{T \supseteq U \supseteq S} z_U^1 \qquad \forall S \subseteq T, \tag{3.6}$$

$$\sum_{e \in \delta_k(S)} x_e^k \geq 2 \sum_{T \supseteq U \supseteq S} (z_U^{k-1} - z_U^k) \qquad \forall S \subseteq T, \; and \; k = 2, \cdots, m-1, \tag{3.7}$$

$$\sum_{e \in \delta_m(S)} x_e^m \geq 2 \sum_{T \supseteq U \supseteq S} z_U^{m-1} \qquad \forall S \subseteq T, \tag{3.8}$$

$$x_e^k \geq 0 \text{ for } e \in E_k, \; where \; k = 1, \cdots, m,$$

$$z_U^k \geq 0 \; \forall U \subseteq T, \; and \; k = 1, \cdots, m-1.$$

A dual of the above LP relaxation can be formulated as follows:

$$C_{dual} = \min 2 \sum_{S \subseteq T} Y_1(S) \tag{3.9}$$

$$\sum_{S: e \in \delta_k(S)} Y_k(S) \leq cost_e^k \qquad \forall e \in E_k, \; \forall k = 1, \cdots, m \tag{3.10}$$

$$\sum_{S \subseteq U} Y_k(S) \leq \sum_{S \subseteq U} Y_{k+1}(S) \quad \forall U \subseteq T, \; \forall k = 1, \cdots, m-1, \tag{3.11}$$

$$Y_k(S) \geq 0 \qquad \forall S \subseteq T, \; \forall k = 1, \cdots, m. \tag{3.12}$$

This dual problem will be used for finding a Heterogeneous Spanning Forest (HSF). The HSF is a collection of m trees where each tree contains a distinct depot. Each target is connected to at least one of the depots and each tree spans a subset of targets and a depot d_k using edges only from E_k.

3.3 Main ideas in the proposed approach

The primal-dual algorithm follows the greedy procedure outlined by Goemans and Williamson in [12]. The basic structure of the algorithm involves maintaining a forest of edges corresponding to each vehicle, and a solution to the dual problem. The edges in the forests are candidates for the set of edges that finally appear in the output (HSF) of the algorithm. Let F_k be the edges in the forest corresponding to the k^{th} vehicle. Initially, F_k is an empty set for each $k \in \{1, \cdots, m\}$. All the components in each F_k are active except the components that contain the depots.

During each iteration of the algorithm, at most one edge is added between two distinct components of some F_k, thus merging the two components in the k^{th} vehicle. The choice of selecting the appropriate edge to be added is based on a dual solution which is also updated during each iteration. Specifically, in each iteration, the algorithm *uniformly* increases the dual variable of each active component by a value ε_{min} that is as large as possible such that none of the constraints in the dual (3.10)-(3.11) are violated. Increasing the dual variable of an active component by ε_{min} will cause one of the following outcomes to happen:

- If one of the constraints in (3.10) becomes tight for some edge $(u, v) \in E_k$ (for $k = 1, \cdots, m$) between two distinct components in F_k, the algorithm adds (u, v) to F_k and merges the two components. If the merged component contains a depot, it becomes inactive; otherwise it is active. If a component $\overline{C}$ of F_k for $k \in \{2, \cdots, m\}$ merges with a component containing d_k, then the total dual cost, $\sum_{S \subseteq \overline{C}} Y_k(S)$, corresponding to $\overline{C}$ serves as an upper bound for $\sum_{S \subseteq \overline{C}} Y_{k-1}(S)$.

- If a constraint in (3.11) becomes tight for a component $\overline{C}$ for some k, then $\overline{C}$ is deactivated in F_k.

The iterative process terminates when all the components become inactive. The final step of the algorithm removes any unnecessary edges that are not required to be in F_k's using a marking procedure that was previously used for the prize-collecting TSP in [12].

3.4 Primal-Dual algorithm for MDHTSP

The initialization, the main loop and the final pruning step of the primal-dual algorithm are presented in **Algorithm 3**. For every $k \in \{1, \cdots, m\}$, let the set of connected components in F_k be denoted by $\mathcal{C}_k$. Initially each $\mathcal{C}_k$ consists of components where each vertex is in its own connected component. For every $k \in \{1, \cdots, m-1\}$ and $\forall C \in \mathcal{C}_k$, the internal variable $w_k(C)$ keeps track of $\sum_{S \subseteq C} Y_k(S)$, i.e., $w_k(C) = \sum_{S \subseteq C} Y_k(S)$. Similarly $\forall C \in \mathcal{C}_k$, $Bound_k(C)$ keeps track of $\sum_{S \subseteq C} Y_{k+1}(S)$. $w_k(C)$ and $Bound_k(C)$ are used to enforce constraints in (3.11). Initially, all the dual variables, $w_k(C)$ and $Bound_k(C)$ are all set to zero. Also for every $k \in \{1, \cdots, m-1\}$ each vertex is $v \in V_k$ is initially unmarked i.e., $mark_k(v)$ is set to zero.

Suppose $k_1 < k_2$ and $k_1, k_2 \in \{1, \cdots, m\}$, then the components in $\mathcal{C}_{k_1}$ tend to merge first because it is cheaper to travel between two targets using the vehicle k_1 than using k_2. For any k, and for components $C_1 \in \mathcal{C}_k$ and $C_2 \in \mathcal{C}_{k+1}$, we define C_1 as the parent of C_2 and C_2 as the child of C_1 if $C_2 \subseteq C_1$. Hence for every $k \in \{1, \cdots, m-1\}$, and for any $C \in \mathcal{C}_k$, we use $Children(C)$ to denote all the children of C present in $\mathcal{C}_{k+1}$. And similarly for every $k \in \{2, \cdots, m\}$ and for any component $C \in \mathcal{C}_k$, $d_k \notin \mathcal{C}_k$, we use $Parent(C)$ to denote the parent component of C in $\mathcal{C}_{k-1}$. According to this definition, $C \in \mathcal{C}_k$ (for any $k = 2, \cdots, m$) does not have a parent if it contains d_k; however, to simplify the presentation, we let $Parent(C)$

Algorithm 3 : *Primal-dual algorithm for MDHTSP*

```
Initialization
$F_k \leftarrow \emptyset$, $\forall k = 1, \cdots, m$;   $\mathcal{C}_k \leftarrow \{\{v\} : v \in V_k\}$, $\forall k = 1, \cdots, m$
for $v \in T$ do
    $mark_k(v) \leftarrow 0$;   $p_k(v) \leftarrow 0$, $\forall k = 1, \cdots, m$
    $w_k(\{v\}) \leftarrow 0$, $\forall k = 1, \cdots, m-1$ ;   $Bound_k(\{v\}) \leftarrow 0$, $\forall k = 1, \cdots, m-1$
    $active_k(\{v\}) \leftarrow 1$, $\forall k = 1, \cdots, m$
    $Children_k(\{v\}) \leftarrow \{v\}$, $\forall k = 1, \cdots, m-1$; $Parent(\{v\}) \leftarrow \{v\}$, $\forall k = 2, \cdots, m$
end for
$active_k(\{d_k\}) \leftarrow 0$, $\forall k = 1, \cdots, m$; $Children_k(\{d_k\}) \leftarrow \emptyset$, $\forall k = 1, \cdots, m-1$; $Parent(\{d_k\}) \leftarrow \emptyset$, $\forall k = 2, \cdots, m$

Main loop
while $\exists C \in \mathcal{C}_1$ such that $active_1(C) = 1$ do
    for $k = 1, \cdots, m$ do
        Find an edge $e_k = (u, v) \in E_k$ with $u \in C_{kx}, v \in C_{ky}$ where $C_{kx}, C_{ky} \in \mathcal{C}_k, C_{kx} \neq C_{ky}$ that minimizes $\varepsilon_k^1 = \frac{(cost_{e_k}^k - p_k(u) - p_k(v))}{active_k(C_{kx}) + active_k(C_{ky})}$
    end for
    for $k = 1, \cdots, m-1$ do
        Let $\mathfrak{C} := \{C : active_k(C) = 1, Children(C) = \emptyset, C \in \mathcal{C}_k\}$. Find $\overline{C} \in \mathfrak{C}$ that minimizes $\varepsilon_k^2 = Bound_k(\overline{C}) - w_k(\overline{C})$
    end for
    $\varepsilon_{min} = \min(\varepsilon_1^1, \cdots, \varepsilon_m^1, \varepsilon_1^2, \varepsilon_{m-1}^2)$
    for $k = 1, \cdots, m$ do
        for $C \in \mathcal{C}_k$ do
            $w_k(C) \leftarrow w_k(C) + \varepsilon_{min} \times active_k(C)$
            For all $v \in C$, $p_k(v) \leftarrow p_k(v) + \varepsilon_{min} \times active_k(C)$
            if $k < m$, then $Bound_k(C) \leftarrow Bound_k(C) + \varepsilon_{min}|Children(C)|$
        end for
    end for
    switch $\varepsilon_{min}$
    Case $\varepsilon_k^1$:
      $F_k \leftarrow F_k \bigcup\{e_k\}$; $\mathcal{C}_k \leftarrow \mathcal{C}_k \bigcup\{C_{kx} \bigcup C_{ky}\} - C_{kx} - C_{ky}$
      $w_k(C_{kx} \bigcup C_{ky}) \leftarrow w_k(C_{kx}) + w_k(C_{ky})$
      if $k < m$, then $Bound_k(C_{kx} \bigcup C_{ky}) \leftarrow Bound_k(C_{kx}) + Bound_k(C_{ky})$
      $Children_k(C_{kx} \bigcup C_{ky}) \leftarrow Children_k(C_{kx}) \bigcup Children_k(C_{ky})$
      For all $C \in Children_k(C_{kx} \bigcup C_{ky})$, $Parent_k(C) \leftarrow C_{kx} \bigcup C_{ky}$
      if $d_k \in C_{kx} \bigcup C_{ky}$, then
         $active_k(C_{kx} \bigcup C_{ky}) = 0$
         if  $k \leq m-1$,  then deactivate every $C \in Descendants(C_{kx} \bigcup C_{ky})$
         if  $k \geq 2$,  then $Parent(C_{kx} \bigcup C_{ky}) \leftarrow \emptyset$
            Let $C \in \{C_{kx}, C_{ky}\}$ such that $d_k \notin C$; $Children(Parent(C)) \leftarrow Children(Parent(C)) - C$
         end if
      else
         if  $k == 1$  then $active_k(C_{kx} \bigcup C_{ky}) = 1$,
         else
            $active_k(C_{kx} \bigcup C_{ky}) \leftarrow 1$, $Parent(C_{kx} \bigcup C_{ky}) \leftarrow Parent(C_{kx})$
            $Children(Parent(C_{kx})) \leftarrow Children(Parent(C_{kx})) \bigcup\{C_{kx} \bigcup C_{ky}\} - C_{kx} - C_{ky}$
         end if
      end if
    Case $\varepsilon_k^2$: $active_k(\overline{C}) \leftarrow 0$; for every $v \in \{v : v \in \overline{C}, mark_k(v) = 0\}$, $mark_k(v) \leftarrow \overline{C}$
    end switch
end while

Pruning Step
for $k = 1, \cdots, m-1$ do
    $V_0' := \emptyset$. $F_k'$ is obtained from $F_k$ as follows: 1) Remove all the edges incident on the vertex set $\overline{V} = V_0' \cup V_1' \cup \cdots \cup V_{k-1}'$ from $F_k$, 2) Further remove as many edges as possible from $F_k$ so that the following properties two properties hold: All the vertices of the set $v \in T \setminus \overline{V}$ in $F_k$ with $mark_k(v) = 0$ are connected to the depot $d_k$ and if any vertex in the set $T \setminus \overline{V}$ with a label $C$ is connected to $d_k$, then any other vertex in $T \setminus \overline{V}$ with a label $C' \supseteq C$ is also connected to the depot $d_k$.
end for
$F_m'$ is obtained from $F_m$ by removing as many edges as possible from $F_m$ such that all the vertices in the set $T \setminus \overline{V}$, where $\overline{V} = V_0' \cup \cdots \cup V_{N-1}'$, is spanned by $F_m'$.
```

be an empty set if C contains d_k. Initially, for any target $v \in T$, $Children(\{v\})$ is assigned to be equal to $\{v\}$ and $Parent(\{v\})$ is assigned to be equal to $\{v\}$. Also, the components that consist of just the depots neither have a parent nor a child.

During each iteration of the main loop, the dual variable corresponding to each of the *active* components is increased by the *same* amount until one of the constraints in (3.10)-(3.11) become tight. This step is implemented using the variables $p_i(u) = \sum_{S:u\in S} Y_i(S)$, $i = 1, \cdots, m$. As long as the targets u and v are not connected in F_i, $\sum_{S:e\in\delta_i(S)} Y_i(S) = p_i(u)+p_i(v)$ for the edge e joining u and v. It follows that the dual variable of the components containing u and v respectively can at most be increased by $\frac{cost^i_e - p_i(u) - p_i(v)}{active_i(C_{ix}) + active_i(C_{iy})}$ where $e = (u, v)$, $u \in C_{ix}$, $v \in C_{iy}$, and $C_{ix}, C_{iy} \in \mathcal{C}_i$. Once the edge $e = (u, v)$ has been added to the forest F_i, the dual cost $\sum_{S:e\in\delta_i(S)} Y_i(S)$ does not increase, and hence the packing constraint corresponding to e in (3.10) will continue to hold.

If a constraint in (3.10) becomes tight for some vehicle k and for some edge $e \in E_k$, F_k is augmented with this new edge and the two components (say C_{kx}, C_{ky} in $\mathcal{C}_k$) connected by e are merged to form a single connected component. The children of the components C_{kx}, C_{ky} now become the children of the resulting component $C_{kx} \cup C_{ky}$. The resulting component becomes inactive if it contains the depot d_k; otherwise, it is active. In the case when the resulting components becomes inactive, all the descendants of the resulting component also become inactive; and say C_{kx} was the active component during the iteration which *did not* contain the depot, the parent of C_{kx} loses C_{kx} as its child (*Refer to lines 28-38 of the algorithm 3*). In the case when the resulting component is active, then the parent of either C_{kx} or C_{ky} is assigned as the parent of the resulting component (If $k = 1$, C_{kx} and C_{ky} do not have a parent and the algorithm just merges the components and activates the merged component). It turns out that due to our assumption about the costs, both C_{kx} and

C_{ky} must be active and must be the children of the same parent as shown in Lemma 3.1 (*Refer to line 39-45 of the algorithm 3*).

Once an active parent $\overline{C}$ in some vehicle k ($k = 1, \cdots, m-1$) loses all its children, $Bound_k(\overline{C})$ specifies the maximum value that can be attained by $w_k(\overline{C})$. Suppose an active component $\overline{C} \in \mathcal{C}_k$ (for some $k = 1, \cdots, m-1$) does not have any children and the increase in the dual variables results in the constraint $w_k(\overline{C}) \leq Bound_k(\overline{C})$ becoming tight, then the algorithm deactivates the component $\overline{C}$ and marks each of the unmarked vertices in the component with $\overline{C}$ (*Refer to lines 46 of the algorithm 3*).

The algorithm terminates the main loop when all the components in $\mathcal{C}_1$ become inactive. In the final step of the algorithm, a tree F'_k for every $k \in \{1, \cdots, m\}$ is obtained from F_k, from $k = 1$ to $m-1$. Let the set of vertices spanned by the tree F'_k be denoted by V'_k. For ease of explanation, we will also assume that $V'_0 = \emptyset$. The tree F'_k is obtained from F_k as follows: 1) Remove all the edges incident on the vertex set $\overline{V} = V'_0 \cup V'_1 \cup \cdots \cup V'_{k-1}$ from F_k, 2) Further remove as many edges as possible from F_k so that the following properties two properties hold:

- All the vertices of the set $v \in T \setminus \overline{V}$ in F_k with $mark_k(v) = 0$ are connected to the depot d_k,
- If any vertex in the set $T \setminus \overline{V}$ with a label C is connected to d_k, then any other vertex in $T \setminus \overline{V}$ with a label $C' \supseteq C$ is also connected to the depot d_k.

Finally, the tree F'_m is obtained from F_m by removing as many edges as possible from F_m such that all the vertices in the set $T \setminus \overline{V}$, where $\overline{V} = V'_0 \cup \cdots \cup V'_{m-1}$, is spanned by F'_m. Figure 3.1, 3.2, 3.3, 3.4, 3.5, and 3.6 are the key snapshots of the primal-dual algorithm for an example of MDHTSP when there exist three depots and nine targets.

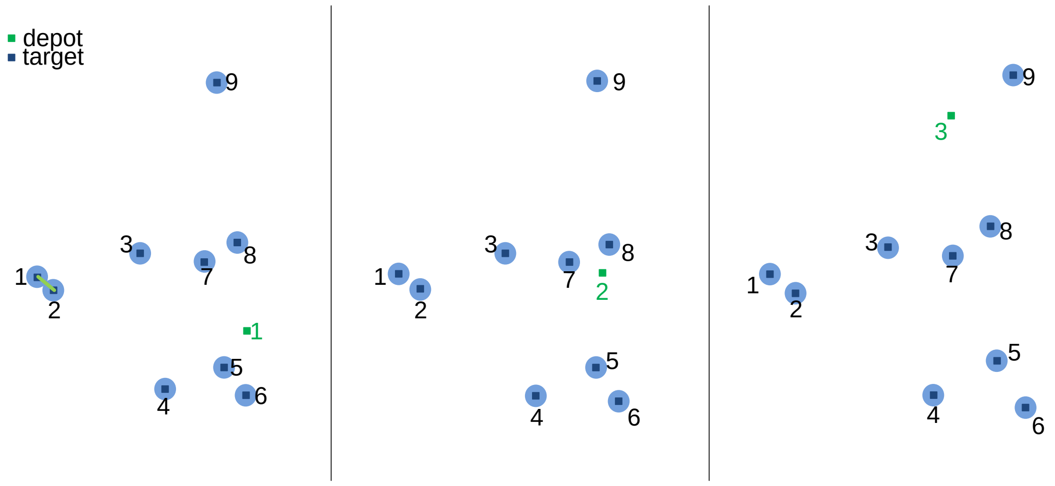

Figure 3.1: Snapshot of the forest for an example with 3 vehicles and 9 targets at the end of the first iteration of the main loop. The radius of the circular region, $p_i(u) := \sum_{S:u\in S} Y_i(S)$, around a target u in the forest F_i is equal to the sum of the dual variables of all the components that contain u in F_i. An edge between target 1 and 2 is added to F_1 as the corresponding constraint of (3.10) became tight.

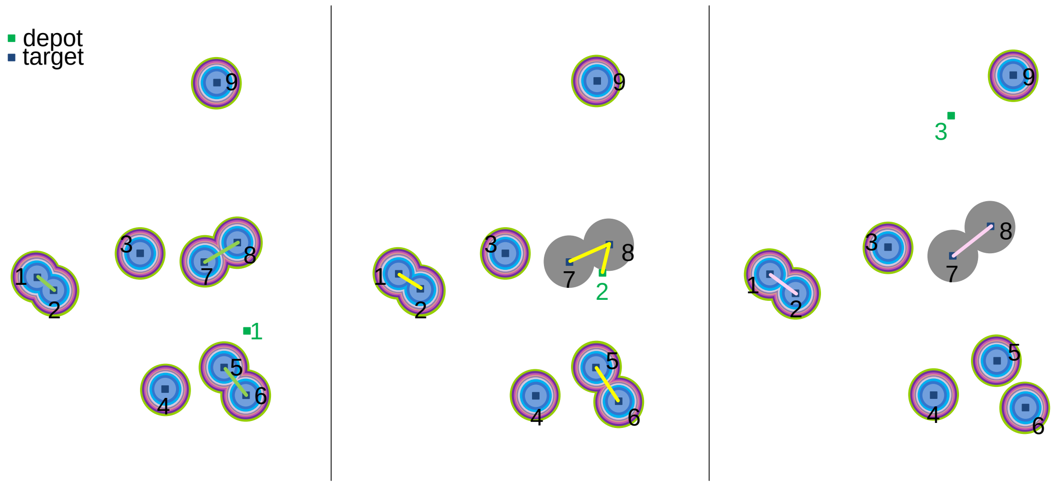

Figure 3.2: Snapshot of the forest after nine iterations of the main loop. The constraint corresponding to the edge joining the target 8 and the depot 2 becomes tight. The edge is added to F_2 and the merged component is deactivated as the target 7 and 8 is now connected to the depot 2. The component $\{7, 8\}$ in F_3 is also deactivated. These components never become active again.

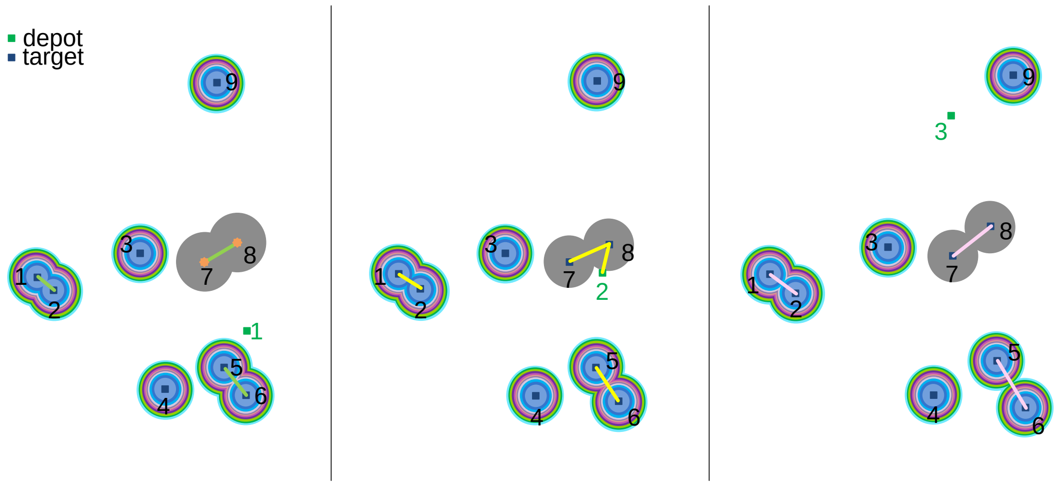

Figure 3.3: Snapshot of the forest after eleven iterations of the main loop. The component $\{7, 8\}$ in F_1 is deactivated and marked since the corresponding constraint in (3.11) became tight.

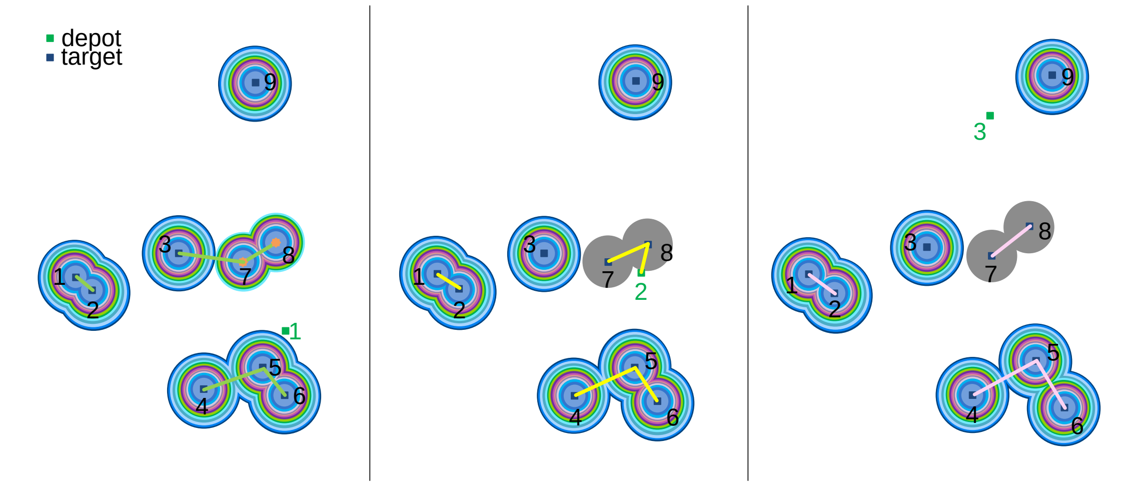

Figure 3.4: Snapshot of the forest after fifteen iterations of the main loop. The edge between the target 3 and 7 is added to F_1 as the corresponding constraint becomes tight. The component $\{7, 8\}$ in F_1 is now merged with the component $\{3\}$ and the new component $\{3, 7, 8\}$ in F_1 is now active again.

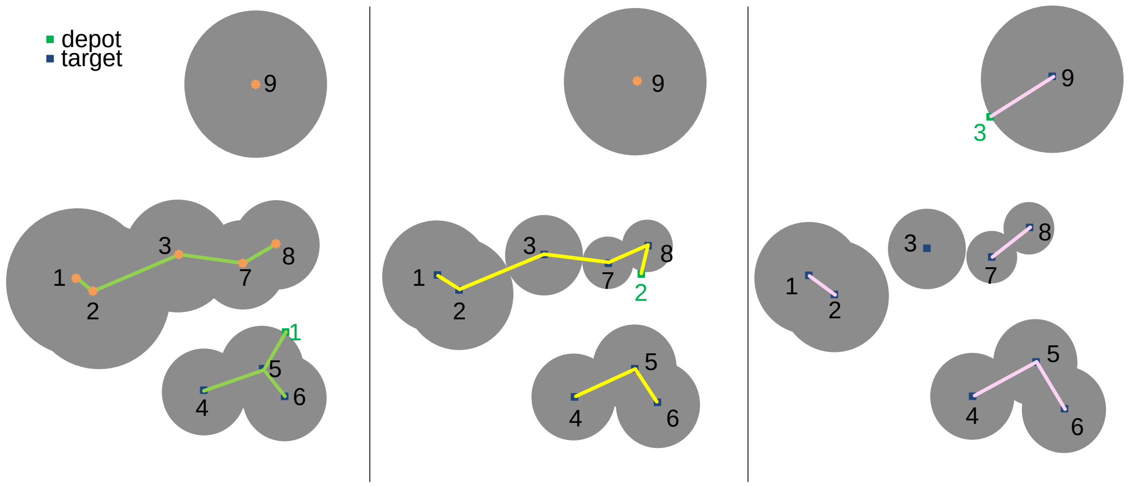

Figure 3.5: Snapshot of the forest at the end of the main loop

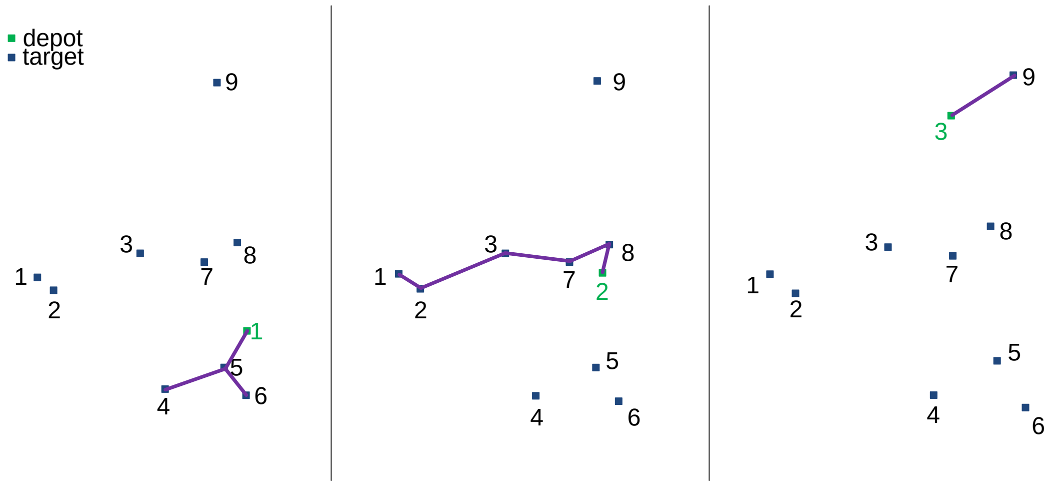

Figure 3.6: Snapshot of the final forest after the pruning step

3.4.1 Properties of the primal-dual algorithm

Consider any target $u \in T$. For any vehicle $k \in \{1, \cdots, m\}$, at the start of the i^{th} iteration, let $C_k^i(u)$ denote the component of F_k containing u.

Lemma 3.1. *The following statements are true for all i and for all $k \in \{1, \cdots, m-1\}$:*

1. $C_{k+1}^i(u) \subseteq C_k^i(u)$ *unless* $C_{k+1}^i(u)$ *contains the depot* d_{k+1}.

2. $active_k(C_k^i(u)) \geq active_{k+1}(C_{k+1}^i(u))$.

Proof. Let us prove this lemma by induction (*Note:* Throughout the proof $k \in \{1, \cdots, m-1\}$). At the start of the first iteration, $C_k^1(u) = C_{k+1}^1(u) = \{u\}$ and the components $C_k^1(u), C_{k+1}^1(u)$ are both active. Therefore the lemmas 3.1.1 and 3.1.2 are correct for $i = 1$. Now let us assume that the lemma is true for the l^{th} iteration for any $l = 1, \cdots, i$. As $active_k(C_k^l(u)) \geq active_{k+1}(C_{k+1}^l(u))$ for any $l = 1, \cdots, i$, it follows that $p_k(u) \geq p_{k+1}(u)$ at the start of the i^{th} iteration.

Proof of Lemma 3.1.1: During the i^{th} iteration, there are three possible cases for the components $C_k^i(u)$ and $C_{k+1}^i(u)$: 1) $C_k^i(u)$ merges with another component in $\mathcal{C}_k$, or, 2) $C_{k+1}^i(u)$ merges with another component in $\mathcal{C}_{k+1}$, or, 3) $C_k^i(u)$ is deactivated because its corresponding constraint in (3.11) becomes tight. It is noted that $C_{k+1}^{i+1}(u)$ will remain as a subset of $C_k^{i+1}(u)$ in the first case. In the third case, $C_k^i(u)$ can be deactivated only when the target u is already connected to the one of the depots in $\{d_{k+1}, \cdots, d_m\}$. Therefore, Lemma 3.1.1 is true by default in the third case.

Let us now examine the second case. If C_{k+1}^i is active and merges with another active component $C_{k+1}^i(v)$ corresponding to target v, we claim that $C_k^i(u) = C_k^i(v)$. Note that

$$\varepsilon_k^1 = \frac{cost_{(u,v)}^k - p_k(u) - p_k(v)}{active_k(C_k^i(u)) + active_k(C_k^i(v))} \leq \frac{cost_{(u,v)}^{k+1} - p_{k+1}(u) - p_{k+1}(v)}{active_{k+1}(C_{k+1}^i(u)) + active_{k+1}(C_{k+1}^i(v))} = \varepsilon_{k+1}^1. \tag{3.13}$$

Therefore, the algorithm will not merge $C_{k+1}^i(u)$ and $C_{k+1}^i(v)$ unless it merges $C_k^i(u)$ and $C_k^i(v)$. If $C_k^i(u) = C_k^i(v)$, it then follows that the merged component $C_{k+1}^i(u) \subseteq C_k^i(u)$.

If $C_{k+1}^i(u)$ is inactive because the target u is connected to one of the depots in $\{d_1, \cdots, d_k\}$, we claim that $C_{k+1}^i(u)$ will never merge with any other component. If this claim is not true and say u is connected to d_k and $C_{k+1}^i(u)$ (which is inactive) merges with some other component $C_{k+1}^i(v)$ corresponding to the target v, then $C_k^i(u) \neq C_k^i(v)$ and $C_{k+1}^i(v)$ must be active. Again from equation (3.13), the algorithm will prefer to merge $C_k^i(u)$ and $C_k^i(v)$ before merging $C_{k+1}^i(u)$ and $C_{k+1}^i(v)$. But once $C_k^i(u)$ and $C_k^i(v)$ are merged, the component $C_{k+1}^i(v)$ becomes inactive, since now v is also connected to the depot d_k so all of $C_{k+1}^i(v), \cdots, C_m^i(v)$ will be also deactivated. Therefore, $C_{k+1}^i(u)$ will remain inactive and will never merge with any other component during the i^{th} iteration. Hence Lemma 3.1.1 is true.

Proof of lemma 3.1.2: If $C^i_{k+1}(u)$ **is inactive**, either $C^i_{k+1}(u)$ must contain the depot d_{k+1} or u is connected to one of the depots in $\{d_1, \cdots, d_k\}$. If $C^i_{k+1}(u)$ already contains d_{k+1}, then $C^{i+1}_{k+1}(u)$ must also be inactive. Therefore, $active_k(C^{i+1}_k(u)) \geq active_{k+1}(C^{i+1}_{k+1}(u)) = 0$. If $C^i_{k+1}(u)$ is inactive because u is already connected to one of the depots in $\{d_1, \cdots, d_k\}$, then by lemma 3.1.1, $C^i_{k+1}(u)$ can never merge with another component during the i^{th} iteration. Therefore, $active_k(C^{i+1}_k(u)) \geq active_{k+1}(C^{i+1}_{k+1}(u))$.

If $C^i_{k+1}(u)$ **is active**, then $active_k(C^i_k(u)) \geq active_{k+1}(C^i_{k+1}(u))$ implies that $C^i_k(u)$ is also active. From Lemma 3.1.1, it follows that $C^i_{k+1}(u) \subseteq C^i_k(u)$. Since the component, $C^i_k(u)$ can never become inactive due to its associated constraint in (3.11) during the i^{th} iteration, it still has at least one active component in F_{k+1} that contains the targets in $C^i_k(u)$, which is $C^i_{k+1}(u)$. The only way $C^i_k(u)$ can lead to an inactive $C^{i+1}_k(u)$ is $C^i_k(u)$ merges with another component containing d_k during the iteration, in which case all the components in $F_{k+1}, \cdots, F_m$, containing the vertices in $C^i_k(u)$, including $C^i_{k+1}(u)$, will be also deactivated. Therefore, $active_k(C^{i+1}_k(u)) \geq active_{k+1}(C^{i+1}_{k+1}(u))$. □

3.4.2 Feasibility and running time analysis

Lemma 3.2. *The primal-dual algorithm produces a feasible heterogeneous spanning forest in polynomial time. Suppose V'_k denotes the set of vertices that are spanned by F'_k for any k, then the sets $V'_1, \cdots, V'_m$ is a disjoint partition of the set of targets T.*

Proof. Using the running time analysis in Goemans and Williamson, one can deduce that our primal-dual algorithm runs in polynomial time.

If we prove that for any $i \in \{1, \cdots, m-1\}$, the vertex sets $V'_1, \cdots, V'_i$ are disjoint, the vertices in set V'_j (where $j = 1, \cdots, i$) are connected to depot d_j, and the vertices in the set $T \setminus (V'_1 \cup \cdots \cup V'_i)$ are connected one of the depots in the set $\{d_{i+1}, \cdots, d_m\}$, then we are done. Let us prove the above claim by induction. For $i = 0$, the claim is trivially satisfied. We shall assume the claim to be true for $i = p$, and prove it holds for $i = p + 1$.

Let $\mathcal{X}_p$ denote the set of vertices that are not spanned by $F'_1 \cup \cdots \cup F'_p$. Based on the marks of each vertex in $\mathcal{X}_p$, $\mathcal{X}_p$ can be partitioned into disjoint, deactivated components $\overline{C}_1, \cdots, \overline{C}_r$ where $\overline{C}_j$ denotes the maximal label of its respective component. Note that F'_p is formed from F_p such that every vertex $v \in T \setminus \{V'_1 \cup \cdots \cup V'_{p-1}\}$ with $mark_p(v) = 0$ remains connected to d_p. The only vertices that are not spanned by F'_p are some of the marked vertices. These vertices were marked because the components in $\mathcal{C}_p$ that contained these vertices were deactivated for making the associated constraints in (3.11) tight. Consider a deactivated component $\overline{C}_q \subseteq \mathcal{X}_p$. $\overline{C}_q$ can be deactivated during an iteration only if $\sum_{S \subseteq \overline{C}_q} Y_p(S) = w_p(\overline{C}_q) = Bound_p(\overline{C}_q) = \sum_{S \subseteq \overline{C}_q} Y_{p+1}(S)$. Also, *during the iteration* when $\overline{C}_q$ becomes deactivated, no target $u \in \overline{C}_q$ is connected to any other target $v \in T \backslash \overline{C}_q$ in F_p. As a result, from lemma 3.1, u does not have any adjacent vertex v in $F_{p+1}, \cdots, F_m$ such that $v \in T \setminus \overline{C}_q$. Since target u is not connected to target $v \in T \setminus \overline{C}_q$ in F_p, u and v cannot be connected in $F_{p+1}, \cdots, F_m$. Therefore, during the construction of F'_{p+1}, the removal of all the edges incident on the vertex set $\overline{V} = V'_1 \cup \cdots \cup V'_p$ does not affect the connectivity of the targets in the set $T \setminus \overline{V}$ to any of the depots in the set $\{d_{p+1}, \cdots, d_m\}$. As a result, all the edges that are incident on any vertex $u \notin \mathcal{X}_p$ can be dropped during the construction of F'_{p+1}. Hence any vertex spanned by the edges in F'_p is not spanned by the edges in F'_{p+1}. $\square$

3.4.3 Approximation ratio analysis

Theorem 3.1. *The proposed algorithm has an approximation ratio of* 2.

Proof. The number of steps required to implement the algorithm depends on the computation of the HSF which can be computed in polynomial time. To prove the approximation ratio we need to show that the cost of the edges in the HSF produced by the primal-dual algorithm is at most equal to the optimal cost of the multiple depot heterogeneous vehicle routing problem (MDHVRP). To prove this, we first simplify the dual cost obtained by the primal dual algorithm. As mentioned previously in lemma 3.2, let $\mathcal{X}_k$ (where $k = 1, \cdots, m-1$) denote the set of vertices that are not spanned by $F_1' \cup \cdots \cup F_k'$. Then, based on the labels of each vertex in $\mathcal{X}_k$, $\mathcal{X}_k$ can be partitioned into finite and disjoint collection of deactivated components; let $|\mathcal{X}_k|$ denote the number of such partitions for each $\mathcal{X}_k$. Let us represent these partitions of $\mathcal{X}_k$ by $\overline{C}_1, \cdots, \overline{C}_{|\mathcal{X}_k|}$ where $\overline{C}_j$ denotes the maximal label of its respective component. Also, let the set $\overline{V}_i$ for every vehicle i denotes the set of vertices that are spanned by the tree $F_0', F_1', \cdots, F_{i-1}'$ ($F_0' = \emptyset$). Then, the dual cost obtained by the primal-dual algorithm is as follows:

$$\begin{aligned}
2\sum_{S\subseteq T} Y_1(S) &= 2\sum_{S\subseteq T, S\not\subseteq \mathcal{X}_1} Y_1(S) + 2\sum_{i=1}^{|\mathcal{X}_1|}\sum_{S\subseteq \overline{C}_i} Y_1(S) \\
&= 2\sum_{S\subseteq V_1'} Y_1(S) + 2\sum_{i=1}^{|\mathcal{X}_1|}\sum_{S\subseteq \overline{C}_i} Y_2(S) \\
&= 2\sum_{S\subseteq T, S\not\subseteq \mathcal{X}_1} Y_1(S) + 2\sum_{S\subseteq T\setminus \overline{V}_2, S\not\subseteq \mathcal{X}_2} Y_2(S) + 2\sum_{i=1}^{|\mathcal{X}_2|}\sum_{S\subseteq \overline{C}_i} Y_2(S) \\
&\vdots \\
&= 2\sum_{k=1}^{m-1}\sum_{S\subseteq T\setminus \overline{V}_k, S\not\subseteq \mathcal{X}_k} Y_k(S) + 2\sum_{i=1}^{|\mathcal{X}_{m-1}|}\sum_{S\subseteq \overline{C}_i} Y_m(S)
\end{aligned}$$

Hence we have the following expression for the dual cost obtained by the primal-dual algorithm:

$$2\sum_{S\subseteq T} Y_1(S) \geq 2\sum_{k=1}^{m-1} \sum_{S\subseteq T\setminus\overline{V}_k, S\nsubseteq \mathcal{X}_k} Y_k(S) + 2\sum_{i=1}^{|\mathcal{X}_{m-1}|} \sum_{S\subseteq \overline{C}_i} Y_m(S) \qquad (3.14)$$

Now we express the cost of the edges in the forest in terms of the dual variables as follows. An edge e is added to F_k (where $k = 1, \cdots, m$) and consequently appears in F'_k only if the corresponding constraint in (3.10) is tight i.e., $\sum_{S:e\in\delta_k(S)} Y_k(S) = cost^k_e$. Therefore, we can express the cost of the edges in the tree F'_k (where $k = 1, \cdots, m-1$) as follows:

$$\begin{aligned}\sum_{e\in F'_k} cost^k_e &= \sum_{e\in F'_k} \sum_{S:e\in\delta_k(S)} Y_k(S) \\ &= \sum_{S\subseteq T} Y_k(S)|F'_k \cap \delta_k(S)|.\end{aligned}$$

For any vehicle $k \in \{1, \cdots, m-1\}$, and for any $S \subseteq \mathcal{X}_k \cup \overline{V}_k$, we have $|F'_k \cap \delta_k(S)| = 0$. This further simplifies the above equation to

$$\sum_{e\in F'_k} cost^k_e \quad = \sum_{S\subseteq T, S\nsubseteq \mathcal{X}_k\cup\overline{V}_k} Y_k(S)|F'_k \cap \delta_k(S)| \quad \forall k \in \{1, \cdots, m-1\} \qquad (3.15)$$

Similarly we can express the cost of the m^{th} tree F'_m in terms of the dual variables as follows; from lemma 3.2, note that F'_m can be decomposed into a set of disjoint sets F'_{mi} where each F'_{mi} consists of edges that form a tree spanning each target from $\overline{C}_i$ ($\overline{C}_1, \cdots, \overline{C}_{|\mathcal{X}_{m-1}|}$ is a the disjoint partition of $\mathcal{X}_{m-1}$) and d_m. An edge e is added to F'_m and consequently appears in F'_{mi} only if the corresponding constraint in (3.10) is tight, $cost^m_e = \sum_{e\in\delta_{mi}(S), S\subseteq\overline{C}_i} Y_m(S)$, where $\delta_{mi}(S)$ consists of all the edges with one end point in S and another end point in $\overline{C}_i \cup \{d_m\} \setminus S$.

$$\sum_{e \in F'_m} cost_e^m \quad = \sum_{S \subseteq \mathcal{X}_{m-1}} Y_N(S)|F'_m \cap \delta_m(S)| = \sum_{i=1}^{|\mathcal{X}_{m-1}|} \sum_{S \subseteq \overline{C}_i} Y_m(S)|F'_{mi} \cap \delta_{mi}(S)| \quad (3.16)$$

Therefore, from equations (3.14), (3.15) and(3.16) the proof of the theorem reduces to showing the following result:

$$\sum_{k=1}^{m-1} \sum_{S \subseteq T, S \not\subseteq \mathcal{X}_k \cup \overline{V}_k} Y_k(S)|F'_k \cap \delta_k(S)| + \sum_{i=1}^{|\mathcal{X}_{m-1}|} \sum_{S \subseteq \overline{C}_i} Y_m(S)|F'_{mi} \cap \delta_{mi}(S)|$$
$$\leq 2 \sum_{k=1}^{m-1} \sum_{S \subseteq V'_k} Y_k(S) + 2 \sum_{i=1}^{|\mathcal{X}_{m-1}|} \sum_{S \subseteq \overline{C}_i} Y_m(S) \quad (3.17)$$

The above result can be proven by induction on the main loop. To see this, let us pick any iteration of the primal dual algorithm. At the start of this iteration, for every vehicle $k \in \{1, \cdots, m-1\}$, let N_a^k be the set of all active components in $\mathcal{C}_k$ such that each active component in this set is not a subset of $\mathcal{X}_k$ and let N_d^k be the set of all inactive components in $\mathcal{C}_k$ such that each inactive component in this set is not a subset of $\mathcal{X}_k$. Note that one of the inactive components of N_d^k must contain the depot d_k. Now consider the m^{th} vehicle; for $i = 1, \cdots, |\mathcal{X}_{m-1}|$, let M_{ai} denote the set of all active components in $\mathcal{C}_m$ such that each active component in this set is a subset of $\overline{C}_i$ ($\overline{C}_1, \cdots, \overline{C}_{|\mathcal{X}_{m-1}|}$ is a the disjoint partition of $\mathcal{X}_{m-1}$). Also, let M_d denote the set of inactive components in $\mathcal{C}_m$ that contain the depot d_m.

Now for $k = 1, \cdots, m-1$, form a graph H_k with components in $N_a^k \cup N_d^k$ as vertices and $e \in F'_k \cap \delta_k(C)$ for $C \in N_a^k \cup N_d^k$ as edges of H_k. H_k is a tree that spans all the vertices in $N_a^k \cup N_d^k$. Similarly form a graph H_{mi} with components in $M_{ai} \cup M_d$ as vertices and $e \in F'_{mi} \cap \delta_m(C)$ for $C \in M_{ai} \cup M_d$ as edges of H_{mi}. H_{mi} is a tree that spans all the vertices in $M_{ai} \cup M_d$.

Let $deg(v, G)$ represent the degree of a vertex v in graph G. During the iteration, the dual variable corresponding to each of the active components is increased by ε_{min}. As a result, the right hand side of the inequality will increase by $2 \cdot \varepsilon_{min} \left(\sum_{k=1}^{m-1} N_a^k + \sum_{i=1}^{|\mathcal{X}_{m-1}|} M_{ai} \right)$, whereas the left hand side of the inequality will increase by $\varepsilon_{min} \left(\sum_{k=1}^{m-1} \sum_{v \in N_a^k} deg(v, H_k) + \sum_{i=1}^{|\mathcal{X}_{m-1}|} \sum deg(v, H_{mi}) \right)$. Hence if we can show that

$$\sum_{k=1}^{m-1} \sum_{v \in N_a^k} deg(v, H_k) + \sum_{i=1}^{|\mathcal{X}_{m-1}|} \sum_{v \in M_{ai}} deg(v, H_{mi}) \leq 2 \left(\sum_{k=1}^{m-1} |N_a^k| + \sum_{i=1}^{|\mathcal{X}_{m-1}|} |M_{ai}| \right), \quad (3.18)$$

then the proof will be complete. To do this, we show that all but one of the leaves of H_k (for $k = 1, \cdots, m-1$) must be active vertices. This result follows from the fact that a component, which does not contain d_k, can become inactive in $\mathcal{C}_k$ only if the constraint associated with this component in (3.11) becomes tight. Therefore, all the vertices in this inactive component must be marked. Also, if vertex v in H_k is a leaf ($deg(v, H_k) = 1$) then pruning all the edges from this inactive component will not disconnect any target with $mark_k(v) = 0$ from d_k . Hence, the pruning step of the algorithm will ensure that an inactive component can never be a leaf vertex in H_k unless it contains d_k. Hence,

$$\sum_{k=1}^{m-1} \sum_{v \in N_d^k} deg(v, H_k) \geq \sum_{k=1}^{m-1} \left(2|N_d^k| - 1 \right). \quad (3.19)$$

We now show the final part of the proof:

$$
\begin{aligned}
&\sum_{k=1}^{m-1}\sum_{v\in N_a^k} deg(v,H_k) + \sum_{i=1}^{|\mathcal{X}_{m-1}|}\sum_{v\in M_{ai}} deg(v,H_{mi}) \\
= &\sum_{k=1}^{m-1}\left(\sum_{v\in N_a^k\cup N_d^k} deg(v,H_k) - \sum_{v\in N_d^k} deg(v,H_k)\right) + \sum_{i=1}^{|\mathcal{X}_{m-1}|}\left(\sum_{v\in M_{ai}\cup M_d} deg(v,H_{mi}) - deg(M_d,H_{mi})\right) \\
\leq &\sum_{k=1}^{m-1}\left(\sum_{v\in N_a^k\cup N_d^k} deg(v,H_k) - \sum_{v\in N_d^k} deg(v,H_k)\right) + \sum_{i=1}^{|\mathcal{X}_{m-1}|}\left(\sum_{v\in M_{ai}\cup M_d} deg(v,H_{mi})\right)
\end{aligned}
$$

For $k = 1, \cdots, m-1$, the tree H_k spans all the vertices in $N_a^k \cup N_d^k$. Therefore, the sum of the degree of all the vertices in H_k is given by $2(|N_a^k| + |N_d^k| - 1)$. Similarly H_{mi} is a tree that spans all the vertices in $M_{ai} \cup M_d$. Therefore, the sum of the degree of all the vertices in H_{mi} is $2|M_{ai}|$. Hence, continuing the proof,

$$
\begin{aligned}
\sum_{k=1}^{m-1}\sum_{v\in N_a^k} deg(v,H_k) + &\sum_{i=1}^{|\mathcal{X}_{m-1}|}\sum_{v\in M_{ai}} deg(v,H_{mi}) \\
&\leq \sum_{k=1}^{m-1}\Big(2\cdot(|N_a^k| + |N_d^k| - 1) - (2|N_d^k| - 1)\Big) + 2\sum_{i=1}^{|\mathcal{X}_{m-1}|}|M_{ai}| \\
&= 2\left(\sum_{k=1}^{m-1}(|N_a^k| - 1) + \sum_{i=1}^{|\mathcal{X}_{m-1}|}|M_{ai}|\right) \\
&< 2\left(\sum_{k=1}^{m-1}|N_a^k| + \sum_{i=1}^{|\mathcal{X}_{m-1}|}|M_{ai}|\right)
\end{aligned}
$$

Hence proved. □

3.5 Simulation results

The proposed primal-dual algorithm was applied to MDHTSP in the procedure of finding a HSF. To generate tours using HSF, we used Christofides algorithm [3] and the Lin-Kernigan Hueristic (LKH) [13]. Although the LKH does not have the performance guarantee, it is considered to be one of the best algorithms for solving a single vehicle TSP. The LKH software is available at http://www.akira.ruc.dk/ keld/research/LKH/ and was used without changing any of its default settings. All the simulations were run on a Dell Precision T5500 workstation (Intel Xeon E5630 processor @ 2.53GHz, 12GB RAM).

For the simulation, the depots and targets were randomly generated in a square area of 1×1 km^2 using a uniform distribution. With the number of vehicles varying from 2 to 5 and the number of targets varying from 15 to 50, several problem instances were created. For each number of vehicles and targets, we generated 50 problem instances. For the purposes of simulating heterogeneity, the minimum turning radius (r) of each vehicle was uniformly increased according to the number of vehicles so that for any two targets $i, j \in T$, $cost^1_{\{i,j\}_1} \leq cost^2_{\{i,j\}_2} \leq, \cdots, \leq cost^m_{\{i,j\}_m}$. Reeds-Shepp path [25] was used as the minimum distance required to travel between any two targets. These travel distances are asymmetric and satisfy the triangle inequality. Figure 3.7 and 3.8 show the example instance results for 2 vehicles and 4 vehicles, respectively.

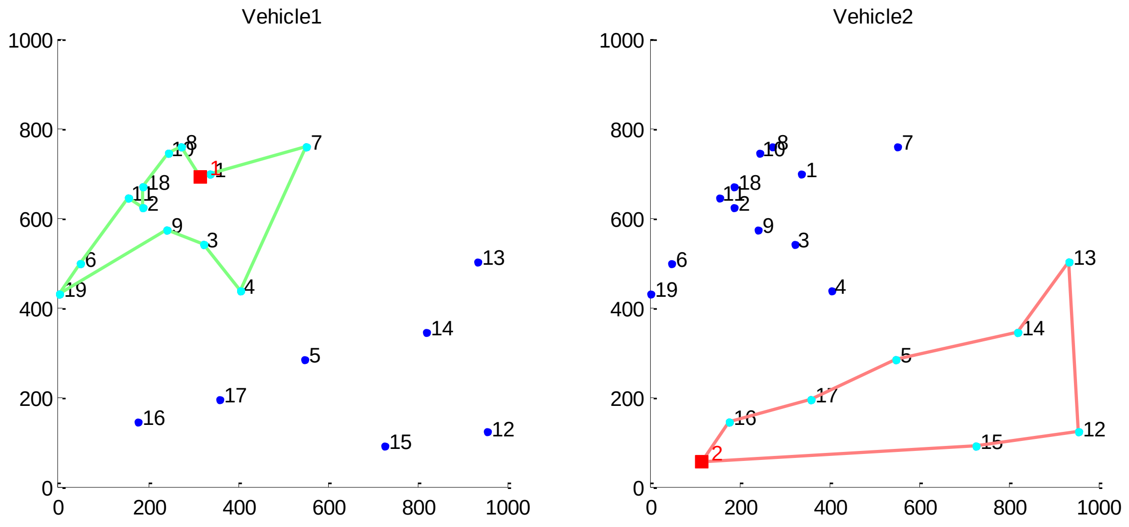

Figure 3.7: Example instance with 2 vehicles

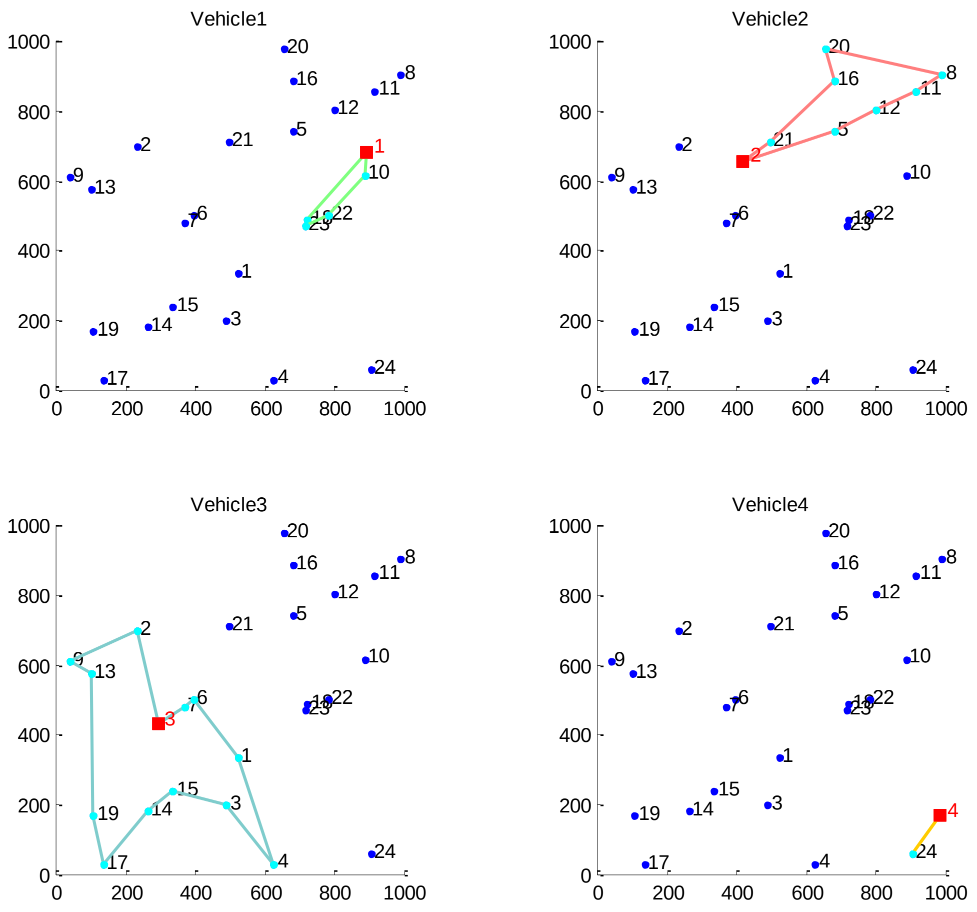

Figure 3.8: Example instance with 4 vehicles

For each instance I, the following equation was used to calculate the approximation ratio of the solution produced by the algorithm for I.

$$ApproximationRatio_I = \frac{Cost_I(Approx)}{Cost_I(DualCost)} \tag{3.20}$$

where,

$Cost_I(Approx)$ = Cost of the primal solution obtained by the algorithm for I,

$Cost_I(DualCost)$ = Cost of the dual problem for the instance I.

The average and maximum approximation ratios of each problem size are shown in Figure 3.9 and Figure 3.10, repectively. Regardless of the problem size, the average approximation ratio was around 1.2 to 1.3 when we used Christofides algorithm, which is slightly higher than when we used LKH. The maximum approximation ratio was lower than 1.5 when we used Christofides algorithm, regardless of the problem size, and was stable compare to LKH. Figure 3.11 shows the average computation time of both Christofides algorithm and LKH for each problem size in seconds. The average computation time was increased as the problem size grew. Generally Christofides algorithm was slightly faster than LKH and the gap between the two was increased as the problem size became larger.

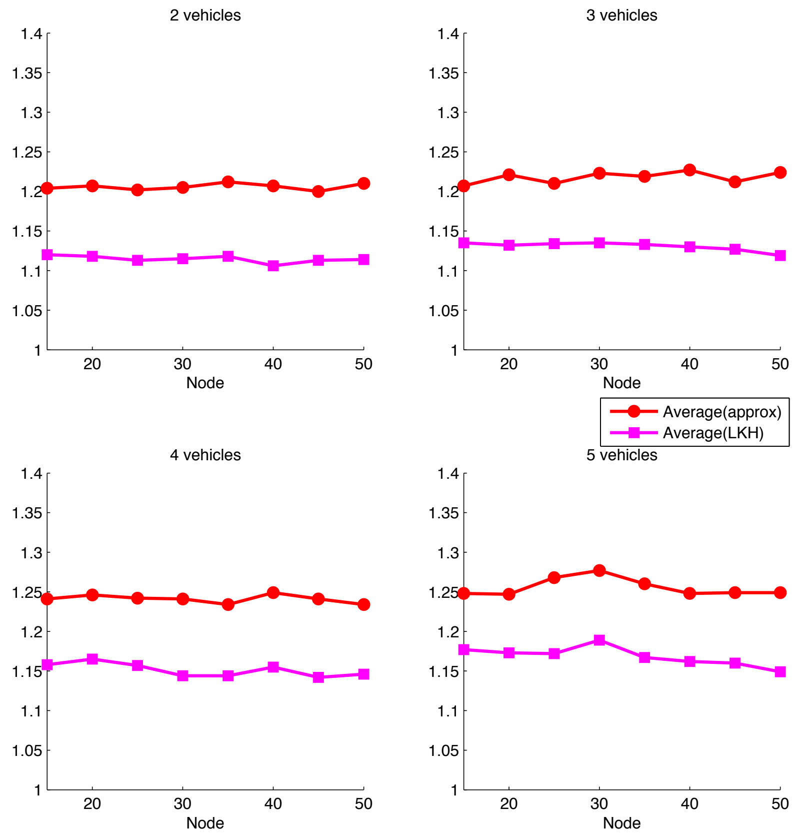

Figure 3.9: Average approximation ratio for 50 instances

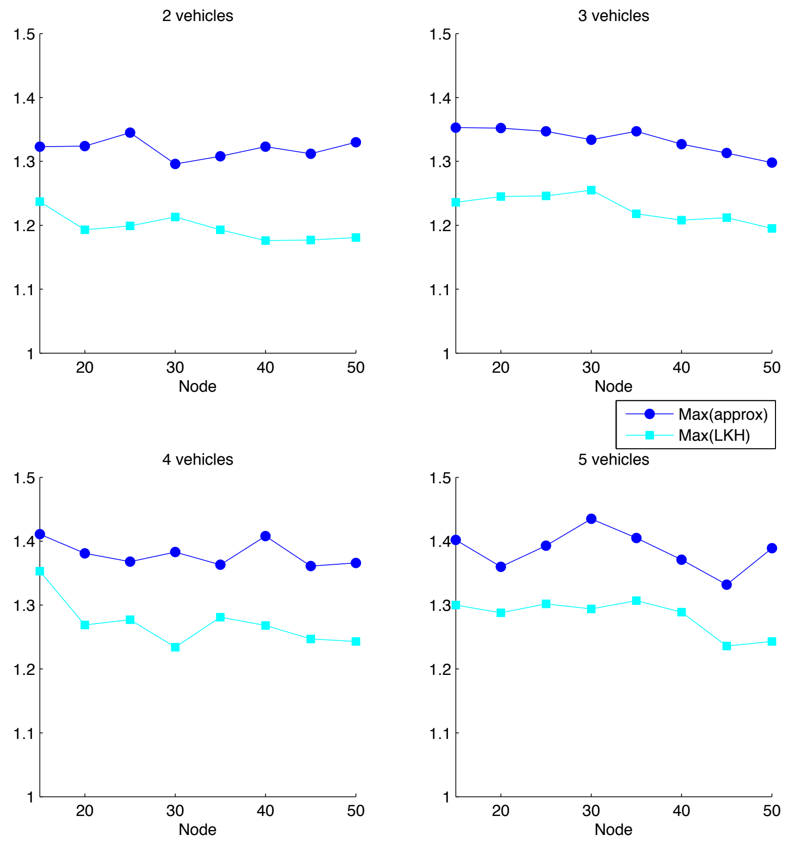

Figure 3.10: Maximum approximation ratio for 50 instances

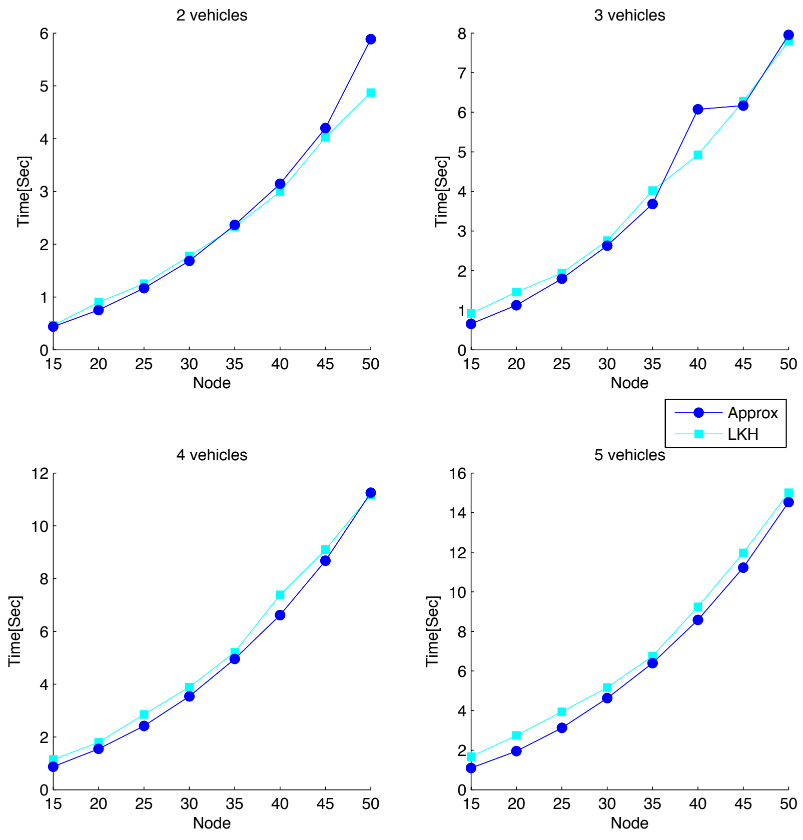

Figure 3.11: Average running time for 50 instances

4. ALGORITHMS FOR A MULTIPLE DEPOT HETEROGENEOUS ASYMMETRIC TRAVELING SALESMAN PROBLEM

In this section, we first state the problem and introduce two algorithms to solve the problem. One is a $m\lceil \log_2(n+1) \rceil$-approximation algorithm where m denotes the number of the vehicles and n denotes the number of targets, and the other is a primal-dual heuristic. We formulate the problem separately for each of the algorithms. In the last section of this section, some computational results are also presented for the proposed algorithms.

4.1 Problem statement

To state the problem, we bring the notations and conditions from the MDHTSP, which we dealt with in section 3. All the notations and conditions we used for the MDHATSP are the same as the ones used for the MDHTSP, except the following ones: Let (u, v) denote the directed edge from vertex u to v. Let E_k be the set of all the directed edges joining any two vertices in V_k. The cost to travel an edge $e \in E_k$ is represented by $cost_e^k$. All the costs are assumed to be positive and satisfy the triangle inequality. We also assumed that $cost_e^1 \leq cost_e^2 \leq \cdots \leq cost_e^m$ for an edge e joining any two targets. The travel costs for each vehicle are assumed to be asymmetric, i.e., the cost of traveling from location i to location j may be different from the cost of traveling location j to location i. For any $S \subset V_k$, let $\delta_k^+(S)$ represent the set of all the edges $(u, v) \in E_k$ such that $u \in S$ and $v \in V_k \setminus S$. Similarly, let $\delta_k^-(S)$ represent the set of all the edges $(u, v) \in E_k$ such that $u \in V_k \setminus S$ and $v \in S$. The objective of the MDHATSP is to find a path for each vehicle such that each target is visited by some vehicle and the sum of the travel costs of all the vehicles is minimized.

4.2 Approximation algorithm using LP relaxation

4.2.1 Problem formulation

In this section, we formulate the problem in the way we can use to develop the approximation algorithm. Let x_e^k denote the binary variable that decides whether edge e is present in the path of the k^{th} vehicle. $x_e^k = 1$ if and only if the edge e is traveled by the k^{th} vehicle and otherwise $x_e^k = 0$. Let ψ_i^k be the binary variable used to assign target i to the k^{th} vehicle. $\psi_i^k = 1$ if target i is assigned to the k^{th} vehicle and otherwise $\psi_i^k = 0$. Let $x := \{x_e^k, e \in E_k, k = 1, \cdots, m\}$ and $\psi = \{\psi_i^k : i \in V_k, k = 1, \cdots, m\}$ denote all the decision variables. A path ($Path_k$) for the k^{th} vehicle is defined by the sequence of vertices visited by the vehicle, *i.e.*, $Path_k := (d_k, u_1, u_2, \cdots, u_{i_k}, d_k)$ where $u_1, \cdots, u_{i_k} \in T$. The cost of traveling $Path_k$ is defined as $cost(Path_k){:=}cost^k_{(d_k,u_1)} + \sum_{j=1}^{i_k-1} cost^k_{(u_j,u_{j+1})} + cost^k_{(u_{i_k},d_k)}$. Now, the MDHATSP can be formulated as follows:

$$C_{opt} = \min \sum_{k=1}^{m} \sum_{e \in E_k} cost_e^k \; x_e^k$$

$$\sum_{k=1}^{m} \psi_i^k = 1 \qquad \forall i \in T, \tag{4.1}$$

$$\sum_{e \in \delta_k^+(\{i\})} x_e^k = \psi_i^k \qquad \forall i \in T, k = 1, \cdots, m, \tag{4.2}$$

$$\sum_{e \in \delta_k^+(\{d_k\})} x_e^k \geq \psi_i^k \qquad \forall i \in T, k = 1, \cdots, m, \tag{4.3}$$

$$\sum_{e \in \delta_k^-(\{i\})} x_e^k = \sum_{e \in \delta_k^+(\{i\})} x_e^k \qquad \forall i \in V_k, k = 1, \cdots, m, \tag{4.4}$$

$$\sum_{e \in \delta_k^+(S)} x_e \geq \psi_i^k \qquad \forall S \subseteq T, i \in S, k = 1, \cdots, m, \tag{4.5}$$

$$\sum_{e \in \delta_k^-(S)} x_e \geq \psi_i^k \qquad \forall S \subseteq T, i \in S, k = 1, \cdots, m, \tag{4.6}$$

$$\psi_i^k \in \{0, 1\} \qquad \forall i \in T, k = 1, \cdots, m, \tag{4.7}$$

$$x_e^k \in \{0, 1\} \qquad \forall e \in E_k, k = 1, \cdots, m. \tag{4.8}$$

The constraints in (4.1) state that each target must be assigned to exactly one vehicle. The out-degree constraints of each target are stated in (4.2). The constraints in (4.3) state that the out-degree of a depot must be at least equal to 1 if any target is assigned to the vehicle corresponding to the depot. The constraints in (4.4) state that the in-degree and out-degree of any vertex must be equal. If target i is assigned to the k^{th} vehicle, then the number of edges in E_k leaving any subset of targets containing i must be at least equal to 1. This connectivity constraint is stated in (4.5). Similarly, if target i is assigned to the k^{th} vehicle, constraints in (4.6) state that the number of edges in E_k entering any subset of targets containing i must be at least equal to 1. The constraints in (4.5),(4.6) are generally referred to as the cut constraints.

We can get the following LP relaxation of the problem by relaxing some constraints.

$$C_{LP*} = \min \sum_{k=1}^{m} \sum_{e \in E_k} cost_e^k \; x_e^k \tag{4.9}$$

$$\sum_{k=1}^{m} \psi_i^k = 1 \qquad \forall i \in T \tag{4.10}$$

$$\sum_{e \in \delta_k^-(\{i\})} x_e^k = \sum_{e \in \delta_k^+(\{i\})} x_e^k \qquad \forall i \in V_k, k = 1, \cdots, m \tag{4.11}$$

$$\sum_{e \in \delta_k^+(S)} x_e \geq \psi_i^k \qquad \forall S \subseteq T, i \in S, k = 1, \cdots, m \tag{4.12}$$

$$\sum_{e \in \delta_k^-(S)} x_e \geq \psi_i^k \qquad \forall S \subseteq T, i \in S, k = 1, \cdots, m \tag{4.13}$$

$$\psi_i^k \geq 0 \qquad \forall i \in T, k = 1, \cdots, m \tag{4.14}$$

$$x_e^k \geq 0 \qquad \forall e \in E_k, k = 1, \cdots, m \tag{4.15}$$

Let this linear program be denoted by *LP**. This will be used in the approximation algorithm that is presented in the following section.

4.2.2 An approximation algorithm for the MDHATSP

The approximation algorithm partitions the targets by solving a LP relaxation of the integer program and then uses the Frieze et al. algorithm[7] to find a path for each vehicle. An approximation algorithm, $approx_3$, is represented in Algorithm 4.

Algorithm 4 $approx_3$

```
1: Solve LP*. Let the optimal solution be denoted by $(x*, \psi*)$.
2: Partition the targets: $\psi_i^k*$ denotes the optimal fraction of the target assigned to
   the $k^{th}$ vehicle in LP*. Assign target $i$ to the $\hat{k}^{th}$ vehicle that corresponds to the
   largest fraction, i.e., $\psi_i^{\hat{k}}* = \max_{l=1}^{m} \psi_i^l*$. Break ties arbitrarily. At the end of this
   step, each target will be assigned to exactly one vehicle. For $k = 1, \cdots, m$, let
   $\mathcal{T}_k$ be the set of targets assigned to the $k^{th}$ vehicle.
3: For $k = 1, \cdots, m$, apply the Frieze et al. single vehicle algorithm to the vertices
   in $\mathcal{T}_k \cup \{d_k\}$ to obtain a path for the $k^{th}$ vehicle.
```

Now, we show that $approx_3$ runs in polynomial time in the following lemma.

Lemma 4.1. *The algorithm approx$_3$ runs in polynomial time.*

Proof. The main steps in *approx$_3$* include solving *LP**, and implementing the Frieze et al. algorithm for each of the vehicles. It is already known that Frieze et al. algorithm runs in polynomial time[7]. Therefore, the proof of the lemma would be complete if *LP** is solvable in polynomial time. The difficulty involved in solving *LP** mainly rests on addressing the cut constraints (4.12),(4.13) as all the other constraints are polynomial in the size of the problem. For all $i \in T$ and $k = 1, \cdots, m$, using the max-flow, min-cut theorem, the cut constraints in (4.12) can be equivalently replaced with flow constraints, which require at least a shipment of ψ_i^k units of commodity from target i to depot d_k. Therefore, for target i and depot d_k, the cut constraints in (4.12) can be replaced with the following set of flow constraints that are polynomial in size:

$$f_{uvi}^k \leq x_{(u,v)}^k \;\forall e = (u, v) \in V_k, \tag{4.16}$$

$$\sum_{v \in T} (f_{uvi}^k - f_{vui}^k) = \begin{cases} \psi_i^k & u = i, \\ 0 & u \notin \{i, d_k\}, \\ -\psi_i^k & u = d_k, \end{cases} \tag{4.17}$$

$$f_{uvi}^k \geq 0 \;\forall u, v \in V^k. \tag{4.18}$$

In the above constraints, f_{uvi}^k denotes the amount of commodity shipped from target i to depot d_k flowing through edge (u, v). These flow constraints can be expressed for each target and depot, and therefore, all the cut constraints in (4.12) can be expressed equivalently using a polynomial number of flow constraints. The same idea can also be applied to the cut constraints in (4.13). Therefore, *LP** can be solved in polynomial time[16]. Hence proved.

□

The following is the main result of this section:

Theorem 4.1. *The approximation ratio of $approx_3$ is $m\lceil \log_2(n+1) \rceil$.*

Proof. For a single vehicle, Asymmetric TSP (ATSP) with n targets and one depot, Williamson[28] showed that the cost of the solution produced by the Frieze et al. algorithm is at most $\lceil \log_2(n+1) \rceil$ times the optimal cost of the Held-Karp relaxation of the ATSP. In the context of our problem, this result implies that $cost(Path_k) \leq \lceil \log_2(n+1) \rceil C^k_{relax}$ where $cost(Path_k)$ denotes the cost of the path found for the k^{th} vehicle and C^k_{relax} represents the optimal Held-Karp relaxation cost. If at least one target is assigned to the k^{th} vehicle ($|\mathfrak{T}_k| \geq 1$), this Held-Karp relaxation cost is defined as follows:

$$C^k_{relax} = \min \sum_{e \in E_k} cost^k_e \; x^k_e$$

$$\sum_{e \in \delta^+_k(\{i\})} x^k_e = \sum_{e \in \delta^-_k(\{i\})} x^k_e = 0 \qquad \forall i \in T \setminus \mathfrak{T}_k, \tag{4.19}$$

$$\sum_{e \in \delta^+_k(\{i\})} x^k_e = \sum_{e \in \delta^-_k(\{i\})} x^k_e = 1 \qquad \forall i \in \mathfrak{T}_k \cup \{d_k\}, \tag{4.20}$$

$$\sum_{e \in \delta^+_k(S)} x_e \geq 1 \qquad \forall S \subset \mathfrak{T}_k \cup \{d_k\}, \tag{4.21}$$

$$x^k_e \geq 0 \qquad \forall e \in E_k. \tag{4.22}$$

As the travel costs satisfy the triangle inequality, Nguyen[20] has shown that the degree constraints in the above formulation can be relaxed without changing the optimal cost of the Held-Karp relaxation as follows:

$$C^k_{relax} = \min \sum_{e \in E_k} cost^k_e \; x^k_e$$

$$\sum_{e \in \delta^-_k(\{i\})} x^k_e = \sum_{e \in \delta^+_k(\{i\})} x^k_e \qquad \forall i \in V_k, \tag{4.23}$$

$$\sum_{e \in \delta^+_k(S)} x_e \geq 1 \qquad \forall S \subset \mathfrak{T}_k \cup \{d_k\}, \tag{4.24}$$

$$x^k_e \geq 0 \qquad \forall e \in E_k. \tag{4.25}$$

Now, we prove that the sum of the optimal cost of the Held-Karp relaxations, $\sum_{k=1}^m C^k_{relax}$, can be bounded by at most m times the optimal cost of $LP*$ defined in (4.9)-(4.15), *i.e.*, $\sum_{k=1}^m C^k_{relax} \leq mC_{LP*}$. Let (x^*, ψ^*) be an optimal solution of LP*. Then, construct a solution $\tilde{x}^k_e := mx^{k*}_e \; \forall e \in V_k, k = 1, \cdots, m$. $\tilde{x}^k_e$ is a feasible solution to the linear program defined in (4.23)-(4.25) due to the following reasons: For all $i \in V_k$,

$$\begin{aligned}\sum_{e \in \delta^-_k(\{i\})} \tilde{x}^k_e &= m \sum_{e \in \delta^+_k(\{i\})} x^{k*}_e \\ &= m \sum_{e \in \delta^-_k(\{i\})} x^{k*}_e \qquad \text{(from equation 4.11)} \\ &= \sum_{e \in \delta^-_k(\{i\})} \tilde{x}^k_e.\end{aligned}$$

For all $S \subseteq \mathfrak{T}_k, |S| \geq 1$,

$$\begin{aligned}
\sum_{e \in \delta_k^+(S)} \tilde{x}_e^k &= m \sum_{e \in \delta_k^+(S)} x_e^{k*} \\
&\geq m\psi_i^{k*} \quad \forall i \in S \qquad \text{(from equation 4.12)} \\
&\geq 1. \qquad \text{(As } \psi_i^{k*} \geq \frac{1}{m} \text{ for any } i \in \mathfrak{T}_k\text{)}
\end{aligned}$$

For all $S \subset \mathfrak{T}_k \cup \{d_k\}, |S \cap \{d_k\}| \geq 1$,

$$\begin{aligned}
\sum_{e \in \delta_k^+(S)} \tilde{x}_e^k &= \sum_{e \in \delta_k^-(V_k \setminus S)} \tilde{x}_e^k \\
&= m \sum_{e \in \delta_k^-(V_k \setminus S)} x_e^{k*} \\
&\geq m\psi_i^{k*} \quad \text{for all } i \in V_k \setminus S \quad \text{(from equation 4.13)} \\
&\geq 1. \quad \text{(As } |\mathfrak{T}_k \setminus S| \geq 1, \text{ there is at least one} \\
&\qquad \text{target } i \in V_k \setminus S \text{ such that } \psi_i^{k*} \geq \frac{1}{m}\text{)}
\end{aligned}$$

Therefore, for $k = 1, \cdots, m$, $\tilde{x}_e^k$ is a feasible solution for the linear program in (4.23)-(4.25). Consequently, $\sum_{k=1}^m C_{relax}^k \leq \sum_{e \in E_k} cost_e^k \ \tilde{x}_e^k = m \sum_{e \in E_k} cost_e^k \ x_e^{k*} = mC_{LP*}$. By putting together all the above results, we have,

$$\begin{aligned}
\sum_{k=1}^m cost(Path_k) &\leq \lceil \log_2(n+1) \rceil \sum_{k=1}^m C_{relax}^k \\
&\leq m \lceil \log_2(n+1) \rceil C_{LP*} \\
&\leq m \lceil \log_2(n+1) \rceil C_{opt}.
\end{aligned}$$

□

4.3 A Primal-dual heuristic algorithm

4.3.1 Problem formulation

In this section, we formulate the problem in a way we can use in the primal-dual heuristic. Let x_e^k denote the binary variable that decides whether edge e is present in the path of the k^{th} vehicle. $x_e^k = 1$ if and only if edge e is traveled by the k^{th} vehicle and otherwise $x_e^k = 0$. For $k = 1, \cdots, m-1$, let z_U^k denote the binary variable that determines the set of targets not visited by vehicles at any of the depots in $\{1, \cdots, k-1\}$. A target u is visited by the k^{th} vehicle if and only if $z_U^{k-1} - z_U^k = 1$. Then, an integer linear program for the MDHATSP can be formulated as follows:

$$C_{IP} = \min \sum_{k=1,\cdots,m} \sum_{e \in E_k} cost_e^k \; x_e^k \tag{4.26}$$

$$\sum_{e \in \delta_1^+(S)} x_e^1 + \sum_{T \supseteq U \supseteq S} z_U^1 \geq 1 \qquad \forall S \subseteq T, \tag{4.27}$$

$$\sum_{e \in \delta_k^+(S)} x_e^k \geq \sum_{T \supseteq U \supseteq S} (z_U^{k-1} - z_U^k) \qquad \forall S \subseteq T, \; k = 2, \cdots, m-1, \tag{4.28}$$

$$\sum_{e \in \delta_m^+(S)} x_e^n \geq \sum_{T \supseteq U \supseteq S} z_U^{m-1} \qquad \forall S \subseteq T, \tag{4.29}$$

$$\sum_{e \in \delta_1^-(S)} x_e^1 + \sum_{T \supseteq U \supseteq S} z_U^1 \geq 1 \qquad \forall S \subseteq T, \tag{4.30}$$

$$\sum_{e \in \delta_k^-(S)} x_e^k \geq \sum_{T \supseteq U \supseteq S} (z_U^{k-1} - z_U^k) \qquad \forall S \subseteq T, \; k = 2, \cdots, m-1, \tag{4.31}$$

$$\sum_{e \in \delta_m^-(S)} x_e^n \geq \sum_{T \supseteq U \supseteq S} z_U^{m-1} \qquad \forall S \subseteq T, \tag{4.32}$$

$$z_U^k \leq \sum_{R \supseteq U} z_R^{k-1}, \qquad \forall U \subseteq T, \; k = 2, \cdots, m, \tag{4.33}$$

$$\sum_{U \subseteq T} z_U^k \leq 1, \qquad k = 1, \cdots, m-1 \tag{4.34}$$

$$x_e^k \in \{0,1\} \qquad \forall e \in E_k,\ k = 1, \cdots, m$$

$$z_U^k \in \{0,1\} \qquad \forall U \subseteq T,\ k = 1, \cdots, m-1.$$

By relaxing some of the constraints, LP relaxation of the MDHATSP can be written as follows:

$$C_{LP} = \min \sum_{k=1,\cdots,n} \sum_{e \in E_k} cost_e^k \ x_e^k \tag{4.35}$$

$$\sum_{e \in \delta_1^+(S)} x_e^1 + \sum_{T \supseteq U \supseteq S} z_U^1 \geq 1 \qquad \forall S \subseteq T, \tag{4.36}$$

$$\sum_{e \in \delta_k^+(S)} x_e^k \geq \sum_{T \supseteq U \supseteq S} (z_U^{k-1} - z_U^k) \qquad \forall S \subseteq T,\ k = 2, \cdots, m-1, \tag{4.37}$$

$$\sum_{e \in \delta_m^+(S)} x_e^n \geq \sum_{T \supseteq U \supseteq S} z_U^{m-1} \qquad \forall S \subseteq T, \tag{4.38}$$

$$x_e^k \geq 0 \qquad \forall e \in E_k,\ k = 1, \cdots, m,$$

$$z_U^k \geq 0 \qquad \forall U \subseteq T,\ k = 1, \cdots, m-1.$$

A dual problem of the LP relaxation can be formulated as follows:

$$C_{Dual} = \min \sum_{S \subseteq T} Y_1^+(S) \tag{4.39}$$

$$\sum_{S:e\in\delta_k^+(S)} Y_k^+(S) \leq cost_e^k \qquad \forall e \in E_k,\ k = 1,\cdots,m, \tag{4.40}$$

$$\sum_{S\subseteq U} Y_k^+(S) \leq \sum_{S\subseteq U} Y_{k+1}^+(S) \qquad \forall U \subseteq T,\ k = 1,\cdots,m-1, \tag{4.41}$$

$$Y_k^+(S) \geq 0 \qquad \forall S \subseteq T,\ k = 1,\cdots,m. \tag{4.42}$$

Similar to the symmetric problem, this dual problem will be used for finding a Heterogeneous Directed Spanning Forest (HDSF). The HDSF is a collection of m trees, and each tree spans a subset of targets from a depot.

4.3.2 A Primal-dual heuristic algorithm

The initialization, the main loop and the final pruning step of the primal-dual algorithm are presented in Algorithm 5. Similar to a primal-dual algorithm for the symmetric problem in Algorithm 3, for every $k \in \{1,\cdots,m\}$, let F_k denote the edges in the forest corresponding to the k^{th} vehicle, and let the set of connected components in F_k be denoted by $\mathcal{C}_k$. In the algorithm, an active component represents a component that is not reachable from any of the depots and does not have any incoming edges. An inactive component denotes a component that is reachable from one of the depots or has at least one incoming edge. A violated component denotes a component which does not have any incoming edges. Initially, F_k is an empty set for each $k \in \{1,\cdots,m\}$, and each $\mathcal{C}_k$ consists of components where each vertex is in its own connected component. All components containing the targets are active and all components containing depots are inactive at the initialization step. For every $k \in \{1,\cdots,m\}$, each vertex in $v \in V_k$ is initially unmarked and all the dual variables are all set to zero.

During each iteration of the main loop, we first choose an *active violated* component S_1 for the first vehicle and one of its subsets from each $\mathcal{C}_k$ for all $k \in \{2,\cdots,m\}$,

Algorithm 5 : *Primal-dual algorithm for MDHATSP*

```
1: Initialization
2: $F_k \leftarrow \emptyset, \forall k = 1, \cdots, m$;   $\mathcal{C}_k \leftarrow \{\{v\} : v \in V_k\}, \forall k = 1, \cdots, m$
3: for $v \in T$ do
4:     All the vertices are unmarked.
5:     All the dual variables are set to zero.
6:     $active_k(\{v\}) \leftarrow 1, \forall k = 1, \cdots, m$
7: end for
8: $active_k(\{d_k\}) \leftarrow 0, \forall k = 1, \cdots, m$
9: Main loop
10: while there exists any active component in $\mathcal{C}_1, \cdots, \mathcal{C}_m$ do
11:     Choose active violated components $S = \{S_1, S_2, \cdots, S_m\}$ from $\mathcal{C}_k$, $\forall k \in \{1, \cdots, m\}$, such that $S_1 \supseteq S_2 \supseteq \cdots \supseteq S_m$.
12:     Find an edge $e_k \in E_k$ that has the minimum cost among all the incoming edges to the components in $S$.
13:     $F_k \leftarrow \{e_k\} \cup F_k$
14:     if $e_k$ forms a new strongly connected component, and the component is not reachable from the depot $d_k$ then
15:         Let the strongly connected component be an active component.
16:     else if $e_k$ makes any vertex $v \in S_k$ reachable from the depot $d_k$ then
17:         Let the depot and the all vertices that are reachable from the depot be an inactive component.
18:         if $k < m$ then
19:             Deactivate all the subsets of this component in $\mathcal{C}_{k+1}, \cdots, \mathcal{C}_m$.
20:         end if
21:         if $k > 1$ then
22:             Mark all the vertices in the supersets of this component in $\mathcal{C}_1, \cdots, \mathcal{C}_{k-1}$. Deactivate it if the corresponding component consists of all marked vertices.
23:         end if
24:     else
25:         Deactivate $S_k$.
26:     end if
27:     if there is no active violated component in $\mathcal{C}_1$ then
28:         Pick an inactive strongly connected component in $\mathcal{C}_1$, which is not reachable from $d_1$, and check all the incoming edges to the chosen set.
29:         if one of the connected components consists of all unmarked vertices then
30:             Drop the chosen component and let this unmarked component be a new chosen component. Check the incoming edges to the new chosen set.
31:         else
32:             Combine them and check all the incoming edges. Repeat combining marked components until there is no incoming edge to the combined component. Let this new component be active.
33:         end if
34:     end if
35:     if an active violated $C \in \mathcal{C}_k$ set has all inactive subsets in $\mathcal{C}_{k+1}$ but there exists any violated component among them then
36:         Combine some of the subsets in the same way as in line 25-31 and let it be active.
37:     end if
38: end while
39: Pruning Step
40: Let $F = \{F_1, \cdots, F_m\}$ and $e_i$ be the edge that is added to $F$ at $i^{th}$ iteration.
41: for $i = 1$ of iterations, down to 1 do
42:     if $F \setminus e_i$ is feasible then
43:         $F \leftarrow F \setminus e_i$
44:     end if
45: end for
```

$\{S_2, \cdots, S_m\}$, such that $S_1 \supseteq S_2 \supseteq \cdots \supseteq S_m$.

Now, we increase the dual variables of the chosen components $\{S_1, \cdots, S_m\}$ by the same amount until one of the constraints in (4.40) becomes tight. Let the corresponding incoming edge be denoted by $e_k \in E_k$. Add this new edge to its corresponding tree F_k. There are three possible cases that could arise by adding e_k. 1) If a new strongly connected component is formed, and it is not reachable from the depot d_k, then let the new strongly connected component be an active component. 2) If any vertex, which was active at the beginning of the iteration, becomes reachable from the depot d_k, then let the depot and all the vertices that are reachable from the depot be an inactive component. If $k < m$, deactivate all the subsets of this component in $\mathcal{C}_{k+1}, \cdots, \mathcal{C}_m$. If $k > 1$, mark all the vertices in the supersets of this component in $\mathcal{C}_1, \cdots, \mathcal{C}_{k-1}$ and deactivate it if the corresponding component only consist of marked vertices. This marking process is to make sure that the constraints in (4.41) are not violated at any time. 3) Neither 1) nor 2) happens, deactivate S_k.

Before the termination condition is satisfied, if there is no active component without any incoming edge in $\mathcal{C}_1$, at least one of the components which contains unmarked vertices can be combined with a component which consists of marked vertices, and we let it be a new active violated component in $\mathcal{C}_1$. This is true because during an iteration, every inactive component that contains unmarked vertices should have at least one incoming edge, and if it is not reachable from the depot d_1, it should be reachable from at least one of the components with marked vertices. In the algorithm, we do this combining as follows: 1) Pick an inactive component which has at least one incoming edge and contains some unmarked vertices in $\mathcal{C}_1$. 2) Check all the incoming edges to the selected component. 3) If one finds a connected component that consists of all unmarked vertices, drop the selected component and let that connected component be the new selected component. If one finds a component that all of its directly connected components contain marked vertices, combine them and let it to be a new selected component. 4) Repeat 2) and 3) until there is no incoming edge to the new combined component.

If an active violated component $C \in \mathcal{C}_k$, which consists of both the marked and unmarked vertices, has all inactive subsets in $\mathcal{C}_{k+1}$ for some $k \in \{1, \cdots, m-1\}$ and there exists any violated components in them, combine some of the subsets in the same way we did for $C \in \mathcal{C}_1$ as stated above.

The algorithm terminates its main loop when all components become inactive. As the final step of the algorithm, we perform reverse deleting steps to obtain the final HSF. Let e_i be the edge that is added to F at i^{th} iteration of the main loop. In reverse order, i.e., from $i =$the total number of the iterations of the main loop down to 1, if F is feasible without e_i, then remove it from F. Otherwise, leave e_i in F.

Figure 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, and 4.9 show some essential steps of the algorithm with an example of three depots and fourteen targets.

Finally, we prove the feasibility of the algorithm in the following lemma 4.2.

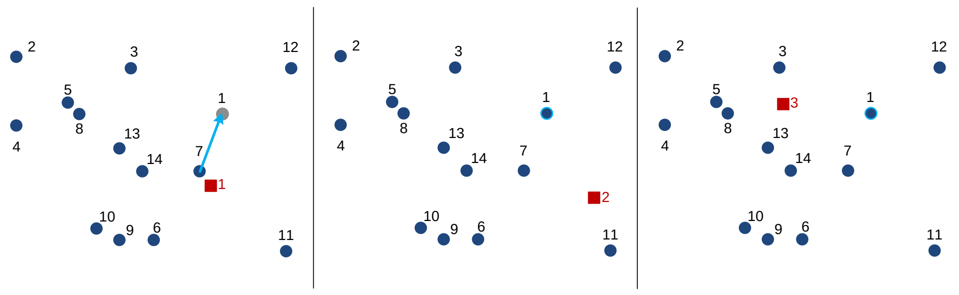
Figure 4.1: Snapshot of the forest for an example with 3 vehicles and 14 targets at the end of the first iteration of the main loop. At the beginning of the first iteration, the components which contain target 1 are chosen to increase the dual variables. An edge coming from target 7 to target 1 is added to F_1 as the corresponding constraint of (4.40) becomes tight. Since target 1 does not form any strongly connected component and is not reachable from d_1, $\{1\} \in \mathcal{C}_1$ is deactivated.

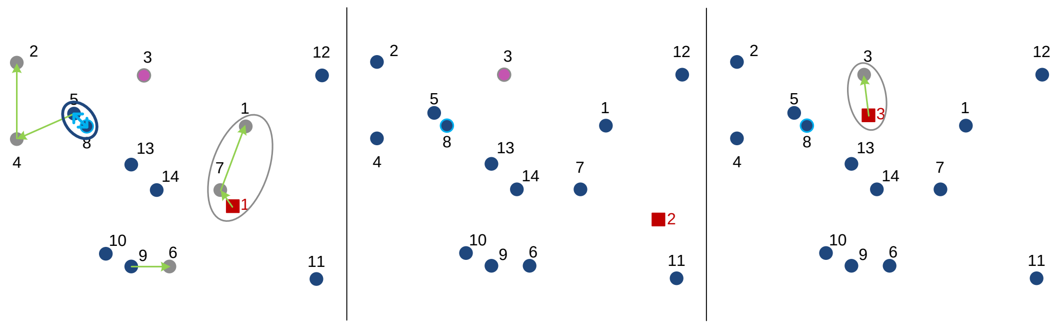
Figure 4.2: Snapshot of the forest after eight iterations of the main loop. The components that contain target 8 are chosen to increase the dual variables and an edge coming from target 5 to target 8 is added to F_1. Since target 5 and target 8 forms a strongly connected component, $\{5, 8\} \in \mathcal{C}_1$ becomes an active component.

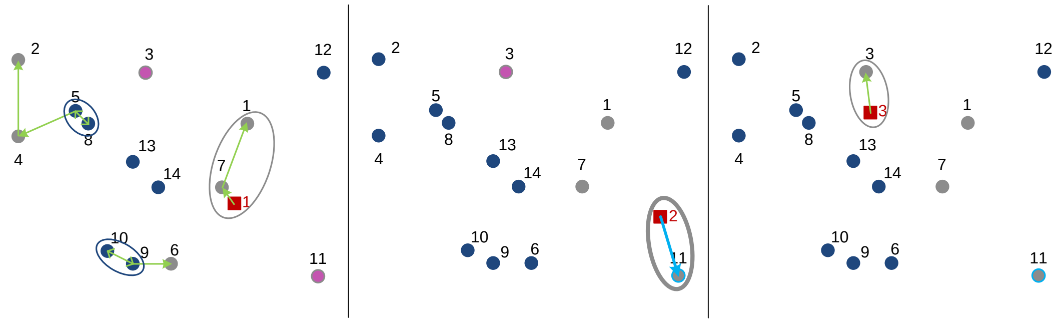

Figure 4.3: Snapshot of the forest after eleven iterations of the main loop. The components that contain target 11 are chosen to increase the dual variables, and an edge coming from depot 2 to target 11 is added to F_2. Since the target 11 is now reachable from d_2, $\{11, d_2\} \in \mathcal{C}_2$ became an inactive component. The subset $\{11\} \in \mathcal{C}_3$ is deactivated and the superset $\{11\} \in \mathcal{C}_1$ is marked and deactivated.

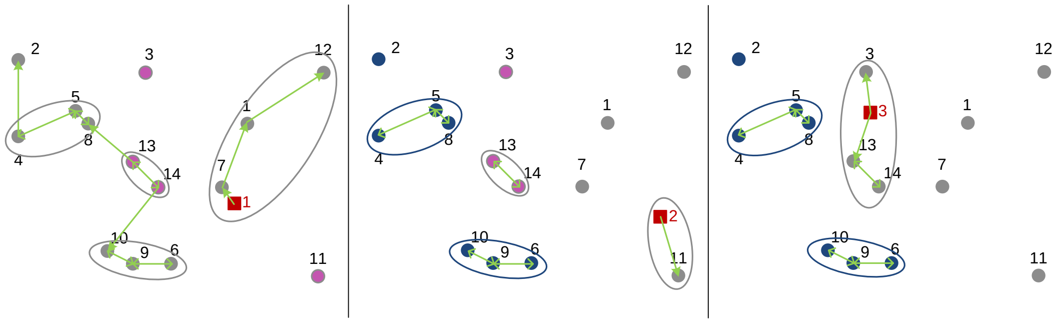

Figure 4.4: Snapshot of the forest in the middle of the thirty ninth iteration of the main loop. In the first map, we can see that the components $\{13, 14\}, \{3\}, \{11\}$ are still violated components but inactive.

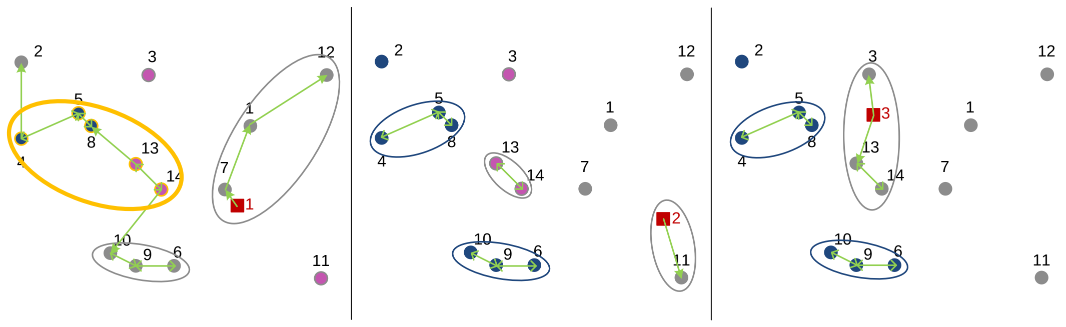

Figure 4.5: Snapshot of the forest after thirty nine iterations of the main loop. By following the combining procedure of the algorithm, now the component $\{4, 5, 8, 13, 14\} \in \mathcal{C}_1$ became an active violated component.

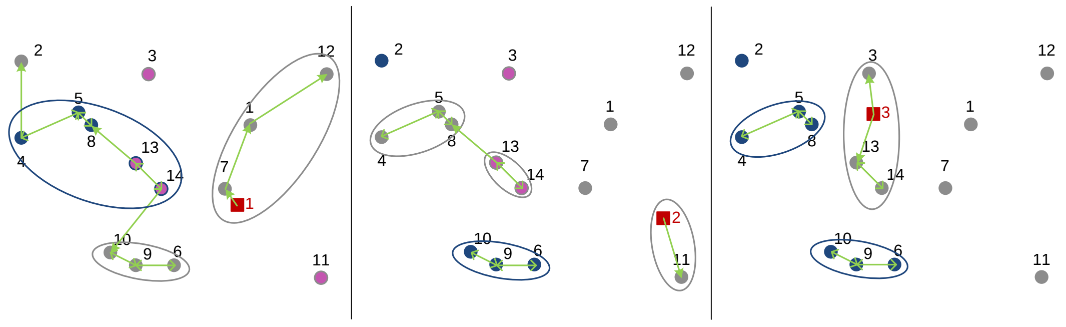

Figure 4.6: Snapshot of the forest in the middle of the forty first iteration of the main loop. The active component $\{4, 5, 8, 13, 14\} \in \mathcal{C}_1$ has all inactive subsets $\{4, 5, 8\}, \{13, 14\} \in \mathcal{C}_2$, but $\{13, 14\} \in \mathcal{C}_2$ is a violated component.

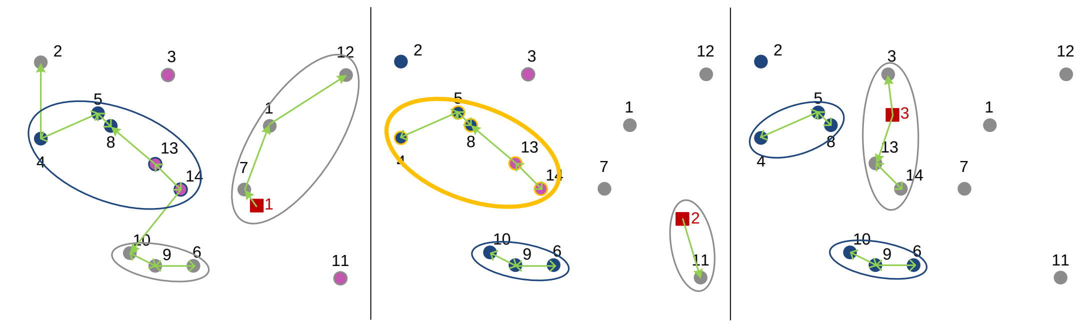

Figure 4.7: Snapshot of the forest after forty one iterations of the main loop. By following the combining procedure of the algorithm, now the component $\{4, 5, 8, 13, 14\} \in \mathcal{C}_2$ became an active violated component.

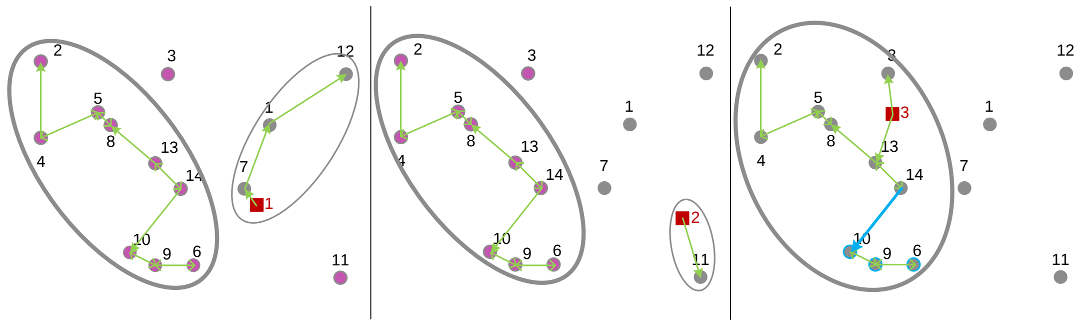

Figure 4.8: Snapshot of the forest after the termination of the main loop.

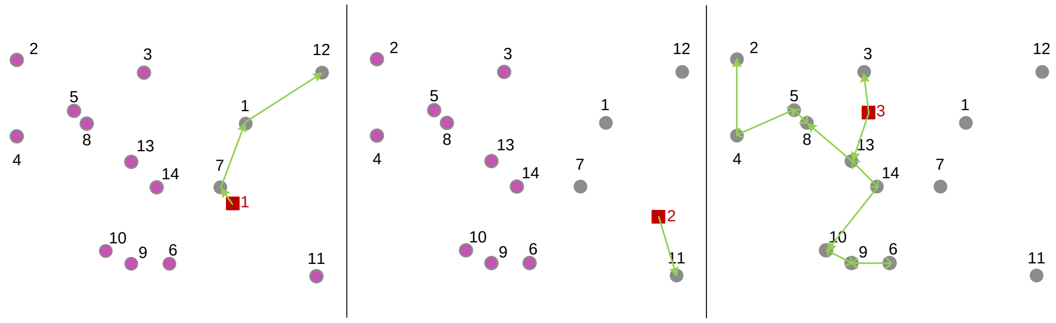

Figure 4.9: Snapshot of the final forest after the pruning step

Lemma 4.2. *The algorithm produces a feasible heterogeneous directed spanning forest, i.e., the trees specified by the collection of edges in $F'_1, \cdots, F'_n$ makes each of the targets reachable from one of the depots. Every vertex appears only once in the forest.*

Proof. The main loop terminates when all components in $\mathcal{C}_1, \cdots, \mathcal{C}_m$ are inactive. A component can be deactivated if one of the following cases occur:

1) The incoming edge added to the component does not form any new strongly connected component, and the vertices in the component are not reachable from any of the depots.

2) The newly added edge makes the component reachable from the depot d_k.

3) One of its subsets or supersets becomes reachable from its depot.

Since the condition 1) only deactivates one of m chosen components, a target in $\mathcal{C}_1, \cdots, \mathcal{C}_m$ cannot be all inactive at the same time by going through only the condition 1). Thus, each target should be deactivated through the condition 2) or 3) at least once to satisfy the termination condition. This implies that each of the targets in T is reachable from at least one of the depots when the main loop is terminated. At the pruning step, the algorithm goes through all the edges in the forest and checks if it is necessary to maintain the feasibility and remove all the unnecessary edges. Therefore, even if a target is connected to the multiple depots during the main loop, it would be connected to only one depot after the pruning step. Hence, the algorithm produces a feasible heterogeneous directed spanning forest. □

4.4 Computational results

Both the the approximation algorithm using LP relaxation in Algorithm 4 and the primal-dual heuristic algorithm in Algorithm 5 have been implemented. In addition to these two, we also implemented a modification of $approx_3$ referred to as $approx_{lkh}$, to improve the performance. In the modification, we solve the LP^* and partition the targets as done in $approx_3$. After partitioning, we use Lin-Kernigan Hueristic (LKH) [13] instead of Frieze et al. algorithm to find a path for each vehicle. Similar to the previous section, the LKH was used without changing any of its default settings to solve the ATSP. All the simulations were run on a Dell Precision T5500 workstation (Intel Xeon E5630 processor @ 2.53GHz, 12GB RAM).

For the simulations, the number of vehicles were varied from 2 to 4 and the number of targets were varied from 20 to 40. All the targets were randomly generated in 5 $\times$ 5 km^2 area using a uniform distribution. For the purposes of simulating heterogeneity, the minimum turning radius (r) of each vehicle was chosen uniformly from the interval $[100, 150]$ meters. The approach, or the heading angle at each target was fixed and randomly chosen from a uniform distribution from the interval $[0, 2\pi]$. Dubins' result[5] was used to calculate the minimum distance required to travel between any two targets. These travel distances are asymmetric and satisfy the triangle inequality.

For a given number of vehicles (m) and targets (n), 50 instances were randomly generated. For a given problem instance I, the bound on the a posterior guarantee provided by an algorithm is defined as $\frac{C^I_{sol_f}}{C^I_{LP*}}$, where $C^I_{sol_f}$ is the cost of the feasible solution found by the implemented algorithm and C^I_{LP*} is the optimal cost of the LP* defined in (4.9).

The average a posterior guarantee of the solutions found by $approx_3$, $approx_{lkh}$, and the primal-dual heuristic are shown in Table 4.1, 4.2, 4.3 along with the theoretical approximation guarantees. The average computation time found by each of the algorithms are shown in Table 4.4, 4.5, 4.6. As we can see from the results, the quality solutions that are, on average, within 1% of the optimum can be found by using $approx_{lkh}$. Also, even though the theoretical a priori guarantee of the approximation algorithm is quite large, the simulation results show that the a posterior guarantee of the solutions found by $approx_3$ was approximately 1.50. These results imply that $approx_3$ finds solutions with bounds that are significantly better than the guarantees indicated by the approximation factor. The quality solutions found by the primal-dual heuristic lie in between $approx_3$ and $approx_{lkh}$. The primal-dual heuristic was the fastest algorithm, and the increasing rate by the problem size is significantly slow compared to $approx_3$ and $approx_{lkh}$.

Table 4.1: Comparison of the theoretical and simulation results for $m = 2$

No. of Targets	Theoretical upper bound ($m\lceil\log_2(n+1)\rceil$)	A Posteriori Bound using $approx_3$	A Posteriori Bound using $approx_{lkh}$	A Posteriori Bound using $primal-dual$
20	8.7846	1.4942	1.0058	1.0495
25	9.4009	1.4516	1.0075	1.0316
30	9.9084	1.4896	1.0061	1.0357
35	10.3399	1.5251	1.0066	1.0230
40	10.7151	1.4830	1.0089	1.0260

Table 4.2: Comparison of the theoretical and simulation results for $m = 3$

No. of Targets	Theoretical upper bound ($m\lceil\log_2(n+1)\rceil$)	A Posteriori Bound using $approx_3$	A Posteriori Bound using $approx_{lkh}$	A Posteriori Bound using $primal-dual$
20	13.1770	1.4714	1.0045	1.0880
25	14.1013	1.4825	1.0068	1.0489
30	14.8626	1.4577	1.0094	1.0453
35	15.5098	1.4897	1.0084	1.0428
40	16.0727	1.4843	1.0102	1.0403

Table 4.3: Comparison of the theoretical and simulation results for $m = 4$

No. of Targets	Theoretical upper bound ($m\lceil\log_2(n+1)\rceil$)	A Posteriori Bound using $approx_3$	A Posteriori Bound using $approx_{lkh}$	A Posteriori Bound using $primal-dual$
20	17.5693	1.4469	1.0046	1.0761
25	18.8018	1.4769	1.0077	1.0784
30	19.8168	1.4477	1.0072	1.0576
35	20.6797	1.5049	1.0085	1.0526
40	21.4302	1.4973	1.0092	1.0628

Table 4.4: Comparison of the computation time in seconds for $m = 2$

No. of Targets	$approx_3$	$approx_{lkh}$	primal-dual heuristic
20	1.648	1.571	0.808
25	5.189	5.157	1.124
30	12.468	12.320	1.465
35	27.478	27.469	1.580
40	52.339	53.776	1.985

Table 4.5: Comparison of the computation time in seconds for $m = 3$

No. of Targets	$approx_3$	$approx_{lkh}$	primal-dual heuristic
20	2.998	3.121	1.044
25	8.373	8.465	1.143
30	19.690	19.875	1.791
35	40.025	40.567	2.037
40	72.348	72.348	2.391

Table 4.6: Comparison of the computation time in seconds for $m = 4$

No. of Targets	$approx_3$	$approx_{lkh}$	primal-dual heuristic
20	3.930	3.871	1.151
25	11.013	11.124	1.457
30	23.968	24.351	1.863
35	45.508	45.795	2.439
40	86.713	86.908	2.763

5. CONCLUSIONS

This dissertation considered three variants/generalizations of a fundamental routing problem involving multiple vehicles and developed approximation algorithms to address each of them. Developing approximation algorithms are difficult for multiple vehicle routing problems because it involves partitioning the targets among the vehicles and then solving a single TSP for each vehicle. This dissertation advances the state of art by developing new algorithms with better approximation factors for each of the considered problems. In particular, a novel 2-approximation algorithm is developed for a multiple terminal, Hamiltonian path problem by using a matroid intersection algorithm. For a special case of this multiple terminal problem where all the vehicles start from the same depot, the dissertation developed a $\frac{5}{3}$-approximation algorithm.

The second routing problem addressed in this dissertation is a multiple vehicle routing problem where the cost of traveling between any two target locations depend on the type of the vehicle. A novel primal-dual algorithm with an approximation ratio of 2 was developed for a special case of this problem where the costs are symmetric, satisfy the triangle inequality and a monotonicity property. This is the first approximation result in the literature that provides a bound independent of the number of vehicles for a multiple depot, heterogeneous vehicle problem. Simulation results were also presented to corroborate the performance of the proposed approximation algorithm.

Finally, we develop the first approximation algorithm for a multiple, heterogeneous vehicle routing problem where the costs are asymmetric and satisfy the triangle inequality. The approximation ratio of this algorithm is $m\lceil \log_2(n+1) \rceil$ where m is the number of vehicles and n is the number of targets. A primal-dual heuristic was also developed and the performance of all the proposed algorithms were corroborated using simulation results.

The following are the fundamental problems that are *open* in the area of approximation algorithms for multiple vehicle routing problems.

- Currently, there is no constant factor approximation algorithm for a general multiple depot, heterogeneous vehicle routing problem when all the travel costs satisfy the triangle inequality. Due to the difficulty involved in addressing heterogeneous costs, it is even possible that no constant factor approximation algorithms are possible. Any contribution is answering this question will be useful.

- Another important class of multiple vehicle routing problems includes cases where the vehicles may be homogenous but there may be additional vehicle-target assignments that must be satisfied. These problems arise in applications where the vehicles may carry different types of sensors and due to resource constraints, some targets must be visited only by some vehicles. Even though some special cases of this class of problems has constant factor approximation algorithms, the scenario where there are general vehicle-target constraints is still open.

- Currently, simulation results suggest that assigning targets to vehicles by rounding a tight LP relaxation (section 4) leads to good approximate solutions. However, a better analysis of this rounding technique is required if a good approximation ratio is desired.

REFERENCES

[1] Cornuejols G. Brezovec, C. and F. Glover. A matroid algorithm and its application to the efficient solution of two optimization problems on graphs. *Mathematical Programming*, 42:471–487, 1988.

[2] J. O. Cerdeira. Matroids and a forest cover problem. *Mathematical Programming 66*, pages 403–405, 1994.

[3] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, 1976.

[4] R. Doshi, S. Yadlapalli, S. Rathinam, and S. Darbha. Approximation algorithms and heuristics for a 2-depot, heterogeneous hamiltonian path problem. *International Journal of Robust and Nonlinear Control*, 21(12):1434–1451, 2011.

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

[6] J Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *J.Res. Natl. Bur. Stand.*, 69B:125–130, 1965.

[7] Alan M Frieze, Giulia Galbiati, and Francesco Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

[8] AM Frieze. An extension of christofides heuristic to the¡ i¿ k¡/i¿-person travelling salesman problem. *Discrete Applied Mathematics*, 6(1):79–83, 1983.

[9] J. E. Davis M. Holland G. L. Feithans, A. J. Rowe and L. Berger. Vigilant spirit control station (vscs) the face of counter. In *Proc. AIAA Guidance, Navigation and Control Conf. Exhibition*, Honululu, Hawaii, United States, 2008. AIAA.

[10] H.N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, 1990.

[11] K. Geiser, D. Slack, E. Allred, and K. Stange. Irrigation scheduling using crop canopyair temperature differences. *Transactions of the American Society of Agricultural and Biological Engineers*, 25(3):689–694, 1982.

[12] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, April 1995.

[13] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[14] J. Hoogeveen. Analysis of christofidesí heuristic: Some paths are more difficult than cycles. *Operations Research Letters*, (10):291–295, 1991.

[15] R. Jackson, S. Idso, R. Reginato, and P. Pinter. Canopy temperature as a crop water stress indicator. *Water Resources Research*, 17(4):1133–1138, 1981.

[16] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[17] Markakis V. Kempe D. Keskinocak P. Koenig S. Kleywegt A. Tovey C. Meyerson A. Lagoudakis, M. and S. Jain. Auction-based multi-robot routing. In

Proceedings of the International Conference on Robotics: Science and Systems, pages 343–350, 2005.

[18] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Dover Publications, 2001.

[19] Rathinam S. Malik, W. and S. Darbha. An approximation algorithm for a symmetric generalized multiple depot, multiple travelling salesman problem. *Operations Research Letters*, 35:747–753, November 2007.

[20] Viet Hung Nguyen. A 2log2 (n)-approximation algorithm for directed tour cover. In *Combinatorial Optimization and Applications*, pages 208–218. Springer, 2009.

[21] S. Rathinam and R. Sengupta. Lower and upper bounds for a multiple depot uav routing problem. In *IEEE Control and Decision Conference*, San Diego, California, 2006.

[22] S. Rathinam and R. Sengupta. 3/2-approximation algorithm for variants of a 2-depot, hamiltonian path problem. *Operations Research Letters*, 38:63–68, 2010.

[23] Sengupta R. Rathinam, S. and S. Darbha. A resource allocation algorithm for multi-vehicle systems with non-holonomic constraints. *IEEE Transactions on Automation Sciences and Engineering*, 4(1):98–104, 2007.

[24] Sivakumar Rathinam. *Routing and monitoring algorithms for UAVs.* PhD thesis, University of California, Berkeley, 2007.

[25] JA Reeds and LA Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[26] B. Rodrigues and Z. Xu. A 3/2-approximation algorithm for multiple depot multiple traveling salesman problem. Scandinavian Symposium and Workshops on Algorithm Theory, 2010.

[27] Vijay V. Vazirani. *Approximation Algorithms*. Springer, March 2004.

[28] David Paul Williamson. *Analysis of the Held-Karp heuristic for the traveling salesman problem*. PhD thesis, Massachusetts Institute of Technology, 1990.

[29] S. Yadlapalli, S. Rathinam, and S. Darbha. 3-approximation algorithm for a two depot, heterogeneous traveling salesman problem. *Optimization Letters*, pages 1–12, 2010.

[30] T. Zajkowski, S. Dunagan, and J. Eilers. Small uas communications mission. In *Eleventh Biennial USDA Forest Service Remote Sensing Applications Conference*, Salt Lake City, UT, 2006.