Heuristics for Two Depot Heterogeneous Unmanned Vehicle Path Planning to Minimize Maximum Travel Cost

Jungyun Bae and Woojin Chung *

Department of Mechanical Engineering, Korea University, Seoul 02841, Korea; jungyun.bae@gmail.com
* Correspondence: smartrobot@korea.ac.kr; Tel.: +82-2-3290-3375

Received: 16 April 2019; Accepted: 21 May 2019; Published: 29 May 2019

Abstract: A solution to the multiple depot heterogeneous traveling salesman problem with a min-max objective is in great demand with many potential applications of unmanned vehicles, as it is highly related to a reduction in the job completion time. As an initial idea for solving the min-max multiple depot heterogeneous traveling salesman problem, new heuristics for path planning problem of two heterogeneous unmanned vehicles are proposed in this article. Specifically, a task allocation and routing problem of two (structurally) heterogeneous unmanned vehicles that are located in distinctive depots and a set of targets to visit is considered. The unmanned vehicles, being heterogeneous, have different travel costs that are determined by their motion constraints. The objective is to find a tour for each vehicle such that each target location is visited at least once by one of the vehicles while the maximum travel cost is minimized. Two heuristics based on a primal-dual technique are proposed to solve the cases where the travel costs are symmetric and asymmetric. The computational results of the implementation have shown that the proposed algorithms produce feasible solutions of good quality within relatively short computation times.

Keywords: Multi-Robot Task Allocation; min-max Traveling Salesman Problem; Path Planning; Primal-dual heuristic

1. Introduction

In most applications of monitoring of large environments, the utilization of multiple autonomous unmanned vehicles (UVs) is preferred to improve system efficiency. For example, unmanned aerial vehicles (UAVs) are widely used in crop management [1,2], traffic monitoring [3], and forest management [4,5]. Various types of multiple UVs are widely used in surveillance [6–8], search and rescue applications [9–11], and transportation [12]. In most of these applications, the fundamental subproblem to efficiently operating the multiple UV systems is finding paths for UVs such that each target is visited by one of the UVs at least once while minimizing the final job completion time. When the UVs involved in an application are not identical to each other, differing in either structure or function or both, and located in distinctive initial locations, the problem is known as the min-max Multiple Depot Heterogeneous Traveling Salesman Problem (MDHTSP). When there exists only one UV—the simplest case—the problem becomes a traveling salesman problem (TSP). Thus, min-max MDHTSP is a generalized TSP and NP-hard [13].

The use of multiple UVs is preferred, especially in large environments. To find an optimal solution for these large sized problems, a large amount of computational effort is required. However, in many applications, the exact location of the vehicles and targets may not be known at the time of operation due to the uncertainty or limited capabilities of the sensors mounted on UVs. The research for path planning under sensor related issues are actively on-going [14–17]. For these cases, especially
involving multiple heterogeneous UVs, having a good suboptimal solution within a short computation time could result in a higher efficiency rather than having an optimal solution after several days of computations. The approaches presented in this paper has been developed for such cases. In this study, as an initial step for solving min-max MDHTSP, the algorithms are developed to solve a TSP for systems involving two structurally heterogeneous UVs while focused on task allocation, finding an optimal assignment of targets for each vehicle and their optimal sequence of visit, and on routing, finding an optimal path for each vehicle in a given sequence. The problem is known as the two depot heterogeneous traveling salesman problem (TDHTSP). Based on previous work [18], the algorithm has been extended for solving the min-max objective by iteratively running the primal-dual algorithm for min-sum TDHTSP with some modification. In addition, we have also considered the case when travel costs are asymmetric with the direction of edges.

As research on the MDHTSP and multiple depot heterogeneous vehicle routing problems (MDHVRP) has advanced from a single agent problem, a majority of the research in literature solves the problem for the total travel costs of all agents to be minimized (min-sum). However, minimizing the maximum travel cost is essential in practical applications because it is directly related to the workload balance between agents and the reduction of job completion time. Few publications have dealt with the min-max MDHTSP or min-max MDHVRP. Franceschelli et al. proposed gossip algorithms for a class of heterogeneous multi-vehicle task assignment and routing problems [19]. Prasad and Choi presented an approximation algorithm for dealing with a task allocation problem for heterogeneous agents starting from a single depot [20].

There are also a few publications that solved min-max multiple depot vehicle routing problems of homogeneous vehicles. Stodola developed an algorithm for a min-max MDVRP based on the Ant Colony Optimization theory [21]. In Reference [22], Wang et al. proposed a heuristic that used a linear program (LP)-based approach as an initial solution and improved in further iterations. Some studies addressed the min-sum MDHTSP. Sundar and Rathinam developed an exact algorithm based on a branch-and-cut method while considering both structural and functional heterogeneity in Reference [23]. The transformation method of generalized MDHATSP into an ATSP was proposed in Reference [24]. A special type of problem called close open multi-depot mixed vehicle routing problem was dealt with in Reference [25], and a new mixed integer programming model and a hybrid metaheuristic were presented. In Table 1, a brief review of the related literature is presented. The categories “Heterogeneity(s)” and “Heterogeneity(f)” refer structural heterogeneity and functional heterogeneity, respectively.

| Depots | Heterogeneity(s) | Heterogeneity(f) | Objective | Methodology | Cost |
|--------|------------------|------------------|-----------|-------------|------|
| Two    | Yes              | No               | Min-Max   | Primal-Dual | Symmetric and Asymmetric |
| [18]   | Yes              | No               | min-sum   | primal-dual | symmetric |
| [19]   | Yes              | No               | min-max   | gossip algo | symmetric |
| [20]   | No               | Yes              | min-max   | ant colony algo | symmetric |
| [21]   | No               | Yes              | min-max   | LP based heuristic | symmetric |
| [22]   | No               | No               | min-max   | branch and cut | asymmetric |
| [23]   | Yes              | Yes              | min-sum   | transformation | asymmetric |
| [24]   | Yes              | No               | min-sum   | hybrid GA | symmetric |
| [25]   | No               | Yes              | min-sum   |               | symmetric |

As an initial step of solving min-max MDHTSP in time efficient manner, this paper made the following main contributions. First, the multiple depot heterogeneous traveling salesman problem that involves two heterogeneous vehicles is solved while considering cost symmetricity. Second, the primal-dual method [26] is applied for the min-max objective for the first time. The primal-dual method is an optimization technique that solves mixed integer program (MIP) problems with simple
To apply primal-dual technique, the LP relaxation of MIP should be derived and its dual problem will be introduced. Then, by iteratively varying the dual variables, one can find an optimal or sub-optimal solution in time efficient manner. Lastly, relatively short computational time of the algorithm is shown through an implementation for different sizes of the problem.

The remainder of this paper is structured as follows: In Section 2, the problem is defined. In Section 3, a formulation of the TDHTSP where travel costs are symmetric is presented and a primal-dual based heuristic that solves the problem is proposed. In Section 4, the case where travel costs are asymmetric is considered. The proposed heuristics are implemented and the computational results are presented in Section 5. Section 6 consists some concluding remarks and future scope.

2. Problem Statement

As previously mentioned in Section 1, the Two Depot Heterogeneous Traveling Salesman Problem (TDHTSP) is considered in this article. Since only the structural heterogeneity is considered at this time, we assume that each vehicle has its own distinctive average running velocity and minimum turning radius. To handle this heterogeneity in an efficient manner, we assume that the faster vehicle has a smaller minimum turning radius compared to the slower vehicle. The faster vehicle is indexed as \( r_1 \) and the other one is indexed as \( r_2 \), such that \( \text{cost}_{1}^{ij} \leq \text{cost}_{2}^{ij} \) \( \forall (i, j) \), where \( \text{cost}_{k}^{ij} \) denotes the cost of traveling edge \((i, j)\) using \( r_k \). We also assume that the triangle inequality is satisfied for both UVs.

In this article, two separate algorithms are developed for the cases when the travel costs are symmetric and asymmetric. Initially, the UVs are located at their distinctive depots. Each vehicle starts the route at its depot, visits a set of assigned targets, and finishes the route by turning back to the depot. The objective of this TDHTSP is to find a route for each vehicle such that each target is visited only once by one of the UVs while the maximum final tour cost is minimized.

When a set of \( n \) targets is given, the parameters and decision variables used in formulations, and the variables used in pseudo-codes can be described as follows:

Parameters:
- \( R = \{ r_1, r_2 \} \) a set of UVs
- \( D = \{ d_1, d_2 \} \) a set of depots
- \( T = \{ t_1, \ldots, t_n \} \) a set of targets
- \( V_k = \{ \{ d_k \} \cup T \} \) a set of vertices for \( r_k \)
- \( E_k = \{ (i, j), \forall i, j \in V_k \} \) a set of edges that connect all vertices in \( V_k \)
- \( C_k = \{ \text{cost}_{k}^{ij}, \forall (i, j) \in E_k \} \) a set of travel costs of all edges in \( E_k \)
- \( \delta_k(S) \) the subset of the edges of \( E_k \) that have one end in \( S \) and the other end in \( V_k \setminus S \)
- \( \delta_k^+(S) \) the subset of the edges \((i, j) \in E_k \) such that \( i \in V_k \setminus S \) and \( j \in S \)

Decision variables:
- \( x_{ij}^k \) the decision variable that represents whether edge \((i, j)\) is used for the tour of \( r_k \)
  - \( x_{ij}^k = \begin{cases} 1 & \text{if edge } (i, j) \text{ is traveled by } r_k \\ 0 & \text{otherwise} \end{cases} \)
- \( z_U \) the decision variable that represents partition of targets in \( T \)
  - \( z_U = \begin{cases} 1 & \text{if set } U \text{ contains all the vertices not visited by } r_1 \\ 0 & \text{otherwise} \end{cases} \)
- \( t \) the maximum travel cost

Variables used in algorithm pseudo-codes:
- \( \text{TourCost}_k \) the final tour cost of \( r_k \) within a given partition of the targets
- \( \text{Tour}_k \) the final tour of \( r_k \) within a given partition of the targets
- \( F_k \) a set of edges in the forest corresponding to \( r_k \)
- \( C_k \) a set of connected components in \( F_k \)
- \( Y_k(C) \) the dual variable of a set \( C \) in \( C_k \), which keeps track of \( \sum_{S \subseteq C} Y_k(S) \)

...
3. TDHTSP with Symmetric Travel Costs

3.1. Problem Formulation

When the travel costs are symmetric, i.e., \( cost^k_{ij} = cost^k_{ji} \), \( \forall (i, j) \in E_k, \ k = 1, 2 \), we can formulate an mixed integer program as follows:

\[
C_{IP_1} = \min t
\]

subject to

\[
\sum_{(i,j) \in \delta_k(S)} x^1_{ij} \geq 2 - 2 \sum_{T \supseteq U \supseteq S} z_U \quad \forall S \subseteq T,
\]

(2)

\[
\sum_{(i,j) \in \delta_k(S)} x^2_{ij} \geq 2 \sum_{T \supseteq U \supseteq S} z_U \quad \forall S \subseteq T,
\]

(3)

\[
\sum_{e \in \delta_1(i)} x^1_{ij} = 2 - 2 \sum_{T \supseteq U \supseteq \{i\}} z_U \quad \forall i \in T,
\]

(4)

\[
\sum_{e \in \delta_2(i)} x^2_{ij} = 2 \sum_{T \supseteq U \supseteq \{i\}} z_U \quad \forall i \in T,
\]

(5)

\[
t \geq \sum_{(i,j) \in E_k} cost^k_{ij} x^k_{ij} \quad k = 1, 2,
\]

(6)

\[
x^k_{ij} \in \{0, 1\} \quad \forall i, j \in T, \ k = 1, 2,
\]

(7)

\[
x^k_{ij} \in \{0, 1, 2\} \quad \text{for} \ i = d_k, \ j \in T, \ k = 1, 2,
\]

(8)

\[
z_U \in \{0, 1\} \quad \forall U \subseteq T,
\]

(9)

\[
t \geq 0.
\]

(10)

In this formulation, Equations (2) and (3) are the edge constraints that if there is at least one vertex in \( S \) that is not connected to the other depot, at least two edges must be chosen from \( \delta_k(S) \) for the tour of \( r_k \) for any \( S \subseteq T \). Equations (4) and (5) are the degree constraints of each target. Equation (6) represents that \( t \) is the maximum travel cost, and Equations (7)–(9) are the integer constraints of decision variables. Equation (10) is the nonnegative constraint of \( t \). Now, by relaxing the degree constraints and integer constraints, an LP can be formulated as shown below:

\[
C_{LP_1} = \min t
\]

subject to

\[
\sum_{(i,j) \in \delta_k(S)} x^1_{ij} \geq 2 - 2 \sum_{T \supseteq U \supseteq S} z_U \quad \forall S \subseteq T,
\]

(12)

\[
\sum_{(i,j) \in \delta_k(S)} x^2_{ij} \geq 2 \sum_{T \supseteq U \supseteq S} z_U \quad \forall S \subseteq T,
\]

(13)

\[
t \geq \sum_{(i,j) \in E_k} cost^k_{ij} x^k_{ij} \quad k = 1, 2,
\]

(14)

\[
x^k_{ij} \geq 0 \quad \forall (i, j) \in E_k, \ k = 1, 2,
\]

(15)
Algorithm 2. Initially, each set contains one vertex and each tree \( F \).

The two sets connected through the newly added edge are now combined into a new set. If the new set contains \( d_k \) does not contain the depot, it becomes an active set. Otherwise, the new set is deactivated. If any of the constraints in Equations (19) or (20) becomes tight, the corresponding edge is added to the tree \( T \). The pseudo-codes of the heuristic for TDHTSP is presented in Algorithms 1 and 2. In Algorithm 1, we at first set \( W_1 \) and \( W_2 \) to be equal to 0.5. Based on the initial routing result, we increase or decrease \( W_1 \) with a small value \( c \leq 0.1 \) and set \( W_2 \) as \( 1 - W_1 \) so that the constraint in Equation (22) is tight all the time. With updated \( W_k \) values, the routes are newly derived and we determine whether any reduction exists in \( t \). Iteration stops when either the other vehicle has the maximum tour cost or there exists any violation in the cost condition of \( W_1 \) cost \( \leq W_2 \) cost \( \forall (i, j) \). With fixed values for \( W_1 \) and \( W_2 \), we derive a task allocation for the UVs using the primal-dual heuristic that solves an HSF based on the dual problem in Equations (18)–(24) that is presented in Algorithm 2. Initially, each set contains one vertex and each tree \( F_k \) is empty. The sets containing targets are active, and the sets containing depots are inactive. All dual variables are set to zero, and all vertices are unmarked. In the main loop, the algorithm consistently increases the dual variables of the active sets by an amount that makes one of dual constraints in Equations (19)–(21) tight. If one of the constraints in Equations (19) or (20) becomes tight, the corresponding edge is added to the tree \( F_k \). The two sets connected through the newly added edge are now combined into a new set. If the new set does not contain the depot, it becomes an active set. Otherwise, the new set is deactivated. If \( k = 1 \) and the new set contains \( d_1 \), all children (the subsets in \( C_2 \)) of the newly formed set are also deactivated. If one of the constraints in Equations (21) becomes tight, the corresponding set is deactivated and the vertices in the set are marked. The iteration terminates when all sets in \( C_1 \) become inactive.

At this point, most primal-dual algorithms that solve min-sum TSPs perform reverse-deleting as pruning steps after the main loop is terminated because the edge added at the \( i \)-th iteration has

\[
z_U \geq 0 \quad \forall U \subseteq T, \quad \text{(16)}
\]

\[
t \geq 0 \quad \text{(17)}
\]

To obtain the dual problem of the LP above, we introduce the dual variables, \( Y_1(S) \), \( Y_2(S) \), and \( W_k \) for Equations (12)–(14), respectively. Then, the following dual problem can be introduced.

\[
C_{\text{dual1}} = \max \sum_{S \subseteq T} 2Y_1(S)
\]

subject to

\[
\sum_{S \subseteq E_1(S)} Y_1(S) \leq \text{cost}_i^1 W_1 \quad \forall (i, j) \in E_1, \quad \text{(19)}
\]

\[
\sum_{S \subseteq E_2(S)} Y_2(S) \leq \text{cost}_i^2 W_2 \quad \forall (i, j) \in E_2, \quad \text{(20)}
\]

\[
\sum_{S \subseteq U} Y_1(S) \leq \sum_{S \subseteq U} Y_2(S) \quad \forall U \subseteq T, \quad \text{(21)}
\]

\[
W_1 + W_2 \leq 1 \quad \text{(22)}
\]

\[
Y_k(S) \geq 0 \quad \forall S \subseteq T, k = 1, 2, \quad \text{(23)}
\]

\[
W_k \geq 0 \quad k = 1, 2. \quad \text{(24)}
\]

In the proposed heuristic, the dual variables \( Y_1(S) \), \( Y_2(S) \) can be interpreted as the costs that each set of \( S \) is willing to pay to be connected to the corresponding depots. The dual variable \( W_k \) can be interpreted as the weight that prioritizes one of the UVs, i.e., if \( W_1 > W_2 \), the algorithm will give priority to \( r_2 \), and vice versa. By adjusting the values of \( W_k \) while maintaining the cost condition of \( W_1 \) cost \( \leq W_2 \) cost \( \forall (i, j) \) to be satisfied all times, we may be able to decrease the maximum tour cost in the proposed algorithm. With a fixed \( W_k \), we can obtain a heterogeneous spanning forest (HSF) that connects each vertex to one of the depots by using this dual problem. The obtained HSF is utilized as a target assignment, and the routes can be derived within the partitions using existing algorithms.

3.2. A Heuristic for the TDHTSP

The pseudo-codes of the heuristic for TDHTSP is presented in Algorithms 1 and 2. In Algorithm 1, to reduce the maximum travel cost, we iteratively run Algorithm 2 by varying the values of \( W_1 \) and \( W_2 \). As presented in Algorithm 1, we at first set \( W_1 \) and \( W_2 \) to be equal to 0.5. Based on the initial routing result, we increase or decrease \( W_1 \) with a small value \( c \leq 0.1 \) and set \( W_2 \) as \( 1 - W_1 \) so that the constraint in Equation (22) is tight all the time. With updated \( W_k \) values, the routes are newly derived and we determine whether any reduction exists in \( t \). Iteration stops when either the other vehicle has the maximum tour cost or there exists any violation in the cost condition of \( W_1 \) cost \( \leq W_2 \) cost \( \forall (i, j) \).

With fixed values for \( W_1 \) and \( W_2 \), we derive a task allocation for the UVs using the primal-dual heuristic that solves an HSF based on the dual problem in Equations (18)–(24) that is presented in Algorithm 2.
a smaller cost than the edge added at the $(i+1)$-th iteration for any $i$ in the loop. Deletion of the unnecessary edges in a reverse order would reduce the total sum of the travel costs. However, as we are dealing with a min-max problem, we need to consider the balance of workload. Thus, instead of reverse-deleting, we perform pruning steps we designed in the proposed heuristic. Once the main loop is terminated, we sort the targets into three partitions: $P_1$, $P_2$, and $P_3$. Each $P_k$ for $k = 1, 2$ contains the vertices that are only connected to $d_k$, while $P_3$ contains the vertices that are connected to both $d_1$ and $d_2$. For $P_1$ and $P_2$, we find the minimum spanning trees $T_1$ and $T_2$, respectively. If $P_k$ only contains its depot, $T_k$ is set to be empty. Then, we distribute the targets in $P_k$ to $P_1$ and $P_2$ depending on the current workload. Until all targets in $P_1$ are distributed, the algorithm finds edges $e_k$ for $k = 1, 2$ that connect any vertex in $P_k$ to one of the targets in $P_1$ with the minimum cost. Depending on the current total edge costs in $T_k$, the edge is added to the tree with a lower cost. The corresponding vertex is removed from $P_k$ and added to $P_3$. When $P_k$ becomes empty, each of the targets will be connected to only one of the depots in either $T_1$ or $T_2$. Once partitioning is finished, a tour for each partition can be found using existing algorithms, i.e., LKH [27] in our implementation. In Figure 1, the routes of each iteration of Algorithm 1 for an instance with 120 targets has been presented. Finally, proof of the feasibility of the proposed heuristic for TDHTSP and running time analysis are presented in Lemma 1.

Algorithm 1 A heuristic for min-max TDHTSP

1: $W_1 = W_2 = 0.5$;
2: Determine a heterogeneous spanning forest using the proposed primal-dual heuristic.
3: for $k = 1, 2$ do
4:  Let the connected targets reachable from the depot be a partition and label it as $P_k$.
5: end for
6: for $k = 1, 2$, derive $Tour_{Cost1}$ and $Tour_{k}$ for $r_k$ within $P_k$ (using an existing routing algorithm).
7: $G \leftarrow \max(Tour_{Cost1})$
8: if $Tour_{Cost1} \geq Tour_{Cost2}$ then
9:  while $Tour_{Cost1} \geq Tour_{Cost2}$ do
10:   $W_1 \leftarrow W_1 + e$; $W_2 \leftarrow 1 - W_1$
11:  Redetermine a target assignment using the proposed primal-dual heuristic and obtain the partitions $P'_k$.
12:  for $k = 1, 2$ do
13:   Derive $Tour_{Cost1}'$ and $Tour_{k}'$ within $P'_k$.
14: end for
15: if $Tour'$ is infeasible then
16:  break, [comment: $: W_1 \times cost_{ij}^1 > W_2 \times cost_{ij}^2$ for some $(i, j)$]
17: else
18:  $G' \leftarrow \max(Tour_{Cost1}')$
19:  if $G' < G$ then
20:    $G \leftarrow G'$
21:  for $k = 1, 2$ do
22:   $Tour_{Cost1} \leftarrow Tour_{Cost1}'; Tour_k \leftarrow Tour_k'$
23: end for
24: end if
25: end if
26: end while
27: else
28:  while $Tour_{Cost1} \leq Tour_{Cost2}$ do
29:   $W_1 \leftarrow W_1 - e$; $W_2 \leftarrow 1 - W_1$
30:  Redetermine a target assignment using the proposed primal-dual heuristic and obtain the partitions $P'_k$.
31:  for $k = 1, 2$ do
32:   Derive $Tour_{Cost1}'$ and $Tour_{k}'$ within $P'_k$.
33: end for
34: $G' \leftarrow \max(Tour_{Cost1}')$
35: if $G' < G$ then
36:    $G \leftarrow G'$
37:  for $k = 1, 2$ do
38:   $Tour_{Cost1} \leftarrow Tour_{Cost1}'; Tour_k \leftarrow Tour_k'$
39: end for
40: end if
41: end while
42: end if
43: return $Tour_k, Tour_{Cost_k}$
Algorithm 2: Primal-dual heuristic for finding an HSF

1. Initialization
2. \( F_k \leftarrow \emptyset, C_k \leftarrow \\{v: v \in V_k\} \) for \( k = 1, 2 \)
3. All vertices are unmarked.
4. All dual variables are set to zero.
5. \( \text{active}_k(\{v\}) \leftarrow 1, \forall v \in V_k \) for \( k = 1, 2 \)
6. \( \text{active}_k(\{d_k\}) \leftarrow 0, \) for \( k = 1, 2 \)
7. Main loop
8. while there exists any active component in \( C_1 \) do
   9. for \( k = 1, 2 \) do
   10. Find an edge \( e_k = (i, j) \in E_k \) with \( i \in C_k, j \in C_j \) where \( C_i, C_j \in C_k, C_i \neq C_j \) that minimizes
   11. \[ \epsilon_k = \frac{(\text{min}(\epsilon_i, \epsilon_j) - \text{dual}_k(i) - \text{dual}_k(j))}{\text{active}_k(i) + \text{active}_k(j)}. \]
   12. end for
   13. Let \( R := \{ R : \text{active}_k(R) = 1, \text{One of subsets of R contains } d_k, \forall R \in C_1 \} \).
   14. Find \( R \in R \) that minimizes \( \epsilon_3 = \text{bound}(R) - Y_1(R) \)
   15. \( \epsilon_{\min} = \min(\epsilon_1, \epsilon_2, \epsilon_3) \)
   16. for \( C \in C_k \) do
   17. \( Y_1(C) \leftarrow Y_1(C) + \epsilon_{\min} \times \text{active}_k(C) \)
   18. \( \text{dual}_k(v) \leftarrow \text{dual}_k(v) + \epsilon_{\min} \times \text{active}_k(C), \forall v \in C \)
   19. if \( k = 1 \) then
   20. \( \text{bound}(C) \leftarrow \text{bound}(C) + \epsilon_{\min} \vert \text{Children}(C) \vert \)
   21. end if
   22. end for
   23. end for
   24. if \( \epsilon_{\min} = \epsilon_k \) for \( k = 1 \) or \( 2 \) then
   25. \( F_k \leftarrow \{e_k\} \cup F_k \)
   26. \( C_k \leftarrow C_k \cup \{C_i \cup C_j\} - C_i - C_j \)
   27. \( Y_1(C_i \cup C_j) = Y_1(C_i) + Y_1(C_j) \)
   28. if \( k = 1 \) then
   29. \( \text{bound}(C_i \cup C_j) \leftarrow \text{bound}(C_i) + \text{bound}(C_j) \)
   30. end if
   31. if \( d_k \in \{C_i \cup C_j\} \) then
   32. \( \text{active}_k(C_i \cup C_j) \leftarrow 0 \)
   33. if \( k = 1 \) then
   34. \( \text{active}_1(C) \leftarrow 0 \forall C \in \text{Children}(C_i \cup C_j) \)
   35. end if
   36. else
   37. \( \text{active}_2(C) \leftarrow 1 \)
   38. end if
   39. else
   40. \( \text{active}_1(C) \leftarrow 0 \)
   41. Mark all of the vertices of \( C \) with the label \( \overline{C} \).
   42. end if
43. end while
44. Pruning
45. \( P_1 \leftarrow \) all vertices that are only connected to \( d_k \) for \( k = 1, 2 \)
46. \( P_2 \leftarrow \) the vertices that are connected to both \( d_1 \) and \( d_2 \)
47. Let \( T_2 \) be the minimum spanning tree of \( P_2 \) for \( k = 1, 2 \).
48. Let \( C(T_k) \) be the sum of edge costs present in \( T_k \)
49. while \( P_2 \) is not empty do
50. Find the shortest edge \( e_k \) that connects a vertex in \( P_k \) and a vertex in \( P_k \) for each \( k \).
51. if \( C(T_k) \leq C(T_k) \) then
52. Add \( e_1 \) to \( T_1 \), remove the corresponding vertex from \( P_1 \), and add it to \( P_1 \).
53. else
54. Add \( e_2 \) to \( T_2 \), remove the corresponding vertex from \( P_2 \) and add it to \( P_2 \).
55. end if
56. end while

Lemma 1. The proposed heuristic (in Algorithms 1 and 2) produces a feasible solution for min-max TDHTSP. Each of the vertices is reachable from one of the depots, and every vertex appears only once in the routes. In addition, Algorithm 2 runs in polynomial time.

Proof. During the main loop in Algorithm 2, the loop terminates when all the components of \( C_1 \) become inactive. There exists only two cases that can happen for deactivation. First, a set \( C \) is connected to its depot, and second, one of the constraints in Equation (21) becomes tight. However, the
second case can happen only if the subsets of \( C \) in \( C_2 \) is connected to \( d_2 \). Thus, the only possible way for termination is if each of the targets in \( T \) is connected to either \( d_1 \) or \( d_2 \). Because we make sure that all vertices are connected to one of the components in pruning steps, the proposed heuristic produces a feasible solution for the given problem.

In addition, because there are at most \( n \) components for each robot at any point in time of running Algorithm 2, each iteration will have \( O(n) \) searches, yielding a time bound of \( O(n\log(n)) \) per iteration or \( O(n^2\log(n)) \) for the entire loop. Thus, Algorithm 2 runs in polynomial time.

\[(a)\) Iteration #1–2. \( W_1, W_2 \) are set to 0.5, and \( r_1 \) has the maximum travel cost. Thus, \( W_1 \) is increased by \( \epsilon \), which is set to 0.1 in this example.

\[(b)\) Iteration #3–4. When \( W_1 = 0.52 \) and \( W_2 = 0.48 \), the maximum travel cost has been decreased but \( r_1 \) still has the maximum value; thus, \( W_1 \) was further increased.

\[(c)\) Iteration #5–6. When \( W_1 = 0.54 \) and \( W_2 = 0.46 \), although the balance of workload has been improved, \( r_1 \) still has the maximum value; thus, \( W_1 \) was further increased.

\[(d)\) Iteration #7. When \( W_1 = 0.56 \) and \( W_2 = 0.44 \), \( r_2 \) has the maximum travel cost and iteration is stopped.

**Figure 1.** Routes produced with Algorithm 1 for an instance of 120 targets in a 3000 × 3000 space with symmetric costs: As the routes in (b) have the lowest maximum travel cost, 14,766, it has been chosen for the final routes.
4. TDHTSP with Asymmetric Travel Costs

4.1. Problem Formulation

When the travel costs are asymmetric, i.e., \( \text{cost}_{ij}^k \neq \text{cost}_{ji}^k \), the directions of the edges need to be considered to generate tours. This problem is known as the two depot heterogeneous asymmetric traveling salesman problem (TDHATSP). We can formulate the following LP by considering only the incoming edges.

\[
C_{LP_2} = \min t 
\]

subject to
\[
\sum_{\{i,j\} \in \delta_1^k (S)} x_{ij}^1 \geq 1 - \sum_{T \supseteq U \supseteq S} z_U \quad \forall S \subseteq T, \tag{26}
\]
\[
\sum_{\{i,j\} \in \delta_2^k (S)} x_{ij}^2 \geq \sum_{T \supseteq U \supseteq S} z_U \quad \forall S \subseteq T, \tag{27}
\]
\[
t \geq \sum_{\{i,j\} \in E_k} \text{cost}_{ij}^k x_{ij}^k \quad k = 1, 2, \tag{28}
\]
\[
x_{ij}^k \geq 0 \quad \forall (i,j) \in E_k, \quad k = 1, 2, \tag{29}
\]
\[
z_U \geq 0 \quad \forall U \subseteq T, \quad k = 1, 2. \tag{30}
\]
\[
t \geq 0 \tag{31}
\]

Similar to the symmetric case in Section 3.1, by introducing the dual variables \( Y_1^+(S), Y_2^+(S) \) and \( W_k^+ \) for Equations (26)–(28), the dual problem of the TDHATSP can be formulated as shown below:

\[
C_{dual_2} = \max \sum_{S \subseteq T} Y_1^+(S) \tag{32}
\]

subject to
\[
\sum_{S \in \delta_1^1 (S)} Y_1^+(S) \leq \text{cost}_{ij}^1 W_1^+ \quad \forall (i,j) \in E_1, \tag{33}
\]
\[
\sum_{S \in \delta_1^2 (S)} Y_2^+(S) \leq \text{cost}_{ij}^2 W_2^+ \quad \forall (i,j) \in E_2, \tag{34}
\]
\[
\sum_{S \subseteq U} Y_1^+(S) \leq \sum_{S \subseteq U} Y_2^+(S) \quad \forall U \subseteq T, \tag{35}
\]
\[
W_1^+ + W_2^+ \leq 1 \tag{36}
\]
\[
Y_k^+(S) \geq 0 \quad \forall S \subseteq T, \quad k = 1, 2, \tag{37}
\]
\[
W_k^+ \geq 0 \quad k = 1, 2. \tag{38}
\]

4.2. A Heuristic for the TDHATSP

The difference between the symmetric and asymmetric cases lies in the primal-dual heuristic for obtaining partitions. For asymmetric case, we apply the primal-dual technique to derive a heterogeneous directed spanning forest (HDSF). During iteration to find the best \( W_k^+ \) values, we use the same algorithm presented in Algorithm 1 as we did in Section 3. The algorithm for obtaining an HDSF is presented in Algorithm 3.
Algorithm 3 Primal-dual heuristic for finding an HDSF

1: Initialization
2: $F_1 \leftarrow \emptyset$, $C_k \leftarrow \{(v) : v \in V_k\}$ for $k = 1, 2$
3: All the vertices are unmarked.
4: All the dual variables are set to zero.
5: active$_k\{(v)\} \leftarrow 1$, $\forall v \in V_k$, for $k = 1, 2$
6: active$_k\{(d_k)\} \leftarrow 0$, for $k = 1, 2$
7: Main loop
8: while there exists any active component in $C_1, C_2$ do
9: \hspace{1em} for $k = 1, 2$ do
10: \hspace{2em} Find an edge $e_k = (i, j) \in E_k$ with $i \in C_i, j \in C_j$, where $C_i, C_j \in C_k, C_i \neq C_j$ that minimizes
11: \hspace{2em} $e_k = \frac{w_i j \text{ cost}_{d_k\text{-dual}}(j)}{\text{active}_k(c_k)}$
12: \hspace{2em} Let the corresponding $C_j \in C_k$ be $S_k$, while $S = \{S_1, S_2\}$ satisfies $S_1 \supseteq S_2$ and $S_1, S_2$ are active.
13: \hspace{2em} $P_k \leftarrow \{e_k\} \cup P_k$
14: \hspace{2em} Increase the dual variables of $S_k$ by $e_k$
15: \hspace{2em} if $e_k$ forms a new strongly connected component and the component is not reachable from $d_k$ then
16: \hspace{2em} \hspace{1em} Let the strongly connected component be an active component.
17: \hspace{2em} \hspace{1em} else if $e_k$ makes any vertex $v \in S_1$ reachable from $d_k$ then
18: \hspace{2em} \hspace{2em} \hspace{1em} Let the depot and all vertices that are reachable from the depot be an inactive component.
19: \hspace{2em} \hspace{2em} \hspace{1em} if $e_k = e_1$ then
20: \hspace{2em} \hspace{2em} \hspace{2em} \hspace{1em} Deactivate all subsets of this component in $C_2$.
21: \hspace{2em} \hspace{2em} else
22: \hspace{2em} \hspace{2em} Mark all vertices in the supersets of this component in $C_1$. Deactivate it if the corresponding component consists of all marked vertices.
23: \hspace{2em} end if
24: else
25: \hspace{2em} Deactivate $S_k$.
26: end if
27: if there exists no $S = \{S_1, S_2\}$ that can be chosen that satisfies the given conditions and there exists any inactive set without an incoming edge that is not connected to the depot then
28: \hspace{1em} Pick an inactive component for each $k$ that consists of marked vertices that have incoming or outgoing edges. Combine those connected components until the new component does not have any incoming edges.
29: end if
30: end while
31: Pruning
32: $P_k \leftarrow$ all vertices that are only reachable from $d_k$ for $k = 1, 2$
33: $P_k \leftarrow$ the vertices that are reachable from both $d_1$ and $d_2$
34: Let $T_k$ be the minimum directed spanning tree of $P_k$ for $k = 1, 2$
35: Let $C(T_k)$ be the sum of the edge costs present in $T_k$
36: while $P_k$ is not empty do
37: \hspace{1em} Find the shortest edge $e_k$ that makes a vertex in $P_k$ reachable from the vertices in $P_k$ for each $k$
38: \hspace{2em} if $C(T_1) \leq C(T_2)$ then
39: \hspace{3em} Add $e_1$ to $T_1$, remove the corresponding vertex from $P_1$, and add it to $P_1$
40: \hspace{3em} else
41: \hspace{3em} Add $e_2$ to $T_2$, remove the corresponding vertex from $P_2$, and add it to $P_2$
42: \hspace{3em} end if
43: end while

Similar to the symmetric case, once $W_1$ and $W_2$ are fixed, we derive a target assignment using a primal-dual heuristic that solves an HDSF based on the dual problem in Equations (32)–(38). Initially, each set contains one vertex and each tree $F_k$ is empty. The sets containing targets are active, while the sets containing depots are inactive. All vertices are unmarked, and all dual variables are set to zero. In the main loop, the algorithm iteratively searches for the dual variables that make one of the dual constraints in Equations (33) and (34) tight with the minimum increase, where the corresponding set in $C_1$ and at least one of its subsets is active. The corresponding new edge is added to its tree $F_k$. There are three cases that can occur. First, if the new edge forms any new strongly connected component that is reachable from its depot, let the strongly connected component be a new active component. Second, if the new edge forms a new strongly connected component that is reachable from $d_k$, let the depot and all reachable vertices be a new inactive component. If $e_1$ has the minimum value, all subsets in $C_2$ are deactivated. If $e_2$ is the minimum value, all vertices in the supersets of the component in $C_1$ are marked and deactivated. This step ensures that the third dual constraint in
Equation (35) holds all of the time during iteration. Third, if neither the first nor second case occurs, deactivate the corresponding set. Once the sets are updated, the algorithm checks whether there exists any active set that has at least one active subset that can be chosen. If not, find an inactive set that has no incoming edges and is not reachable from the depot. According to the design of the algorithm, those sets should have at least one connected set that contains marked vertices. Thus, we form a new active component by combining those sets until the new set does not have any incoming edges. Once the main loop is terminated, we perform the pruning steps as we did for the symmetric case in Algorithm 2. Since the edges have directions, we form the minimum directed spanning trees for each partition $P_i$ and iteratively distribute the vertices in $P_i$ to the one that has the lower tree cost. Now, the feasibility of the proposed heuristic for TDHATSP is shown in Lemma 2. Because the running time analysis is very similar with symmetric case, the proof is omitted.

**Lemma 2.** The proposed heuristic (in Algorithm 1 and 3) produces a feasible solution for min-max TDHATSP. Each of the vertices is reachable from one of the depots, and every vertex appears only once in the routes.

**Proof.** During the main loop in Algorithm 3, the loop terminates when all components become inactive. There are three cases that can happen for deactivation: (1) The edge added to the forest does not form any new strongly connected component, and no vertices in the component are reachable from one of the depots; (2) the component has become reachable from its depot; and (3) one of its subsets or supersets becomes reachable from its depot. The vertices would become reachable from one of the depots if (2) or (3) happens at least once at the time of termination.

Assume that only one active component has left for all $C_k$ after having case (1) for all iterations. Then, there exist only two cases that can happen in the next iteration. Either the component becomes reachable from one of the depots and terminates the loop or a new active strongly connected component is formed and continues. This would be true until there exists only one large active component that containing all the vertices left except the depot, which implies the latter case cannot happen at the next iteration. Thus, at least one edge starting from one of the depots should be added before the termination. This implies that, at the time of termination, each vertex is connected to at least one of the depots. During the pruning steps, we make sure that all vertices are connected to only one of the depots while distributing vertices in $P_i$. Thus, even if there existed the vertices that were connected to both depots, only one depot would be left at the end of the algorithm. Hence, the algorithm produces a feasible solution for TDHATSP.

5. Implementation

In this section, we show the computational aspects of the proposed algorithms. All computational experiments were performed with a PC equipped with an Intel® Core™ i5-7600 CPU running at 3.5 GHz with 8 GB RAM. Simulations were performed on a virtual square space with a size of 2000 × 2000 units. The proposed algorithms were implemented using MATLAB software and run repeatedly for instances with varying problem sizes. The number of targets was set to 20, 30, 40, 60, and 80. We evaluated 50 instances for each problem size. The locations of the targets and depots were randomly generated with a uniform distribution. We assumed that both $r_1$ and $r_2$ have the motion of a Reeds–Shepp vehicle [28] for the cost-symmetric cases and the motion of a Dubins vehicle [29] for the cost-asymmetric cases. To ensure that the cost inequality is satisfied, we set the average velocity of movement $v_k$ to be $v_1 > v_2$ and the minimum turning radius $\rho_k$ to be $\rho_1 < \rho_2$. Due to the motion constraints of the robots, we generated the locations of targets to stay within the margins of $\rho_1$. The costs to travel between the locations were set to the minimum travel time between the locations by dividing the Reeds–Shepp curve [30] and Dubins curve [29] by the average velocity of each vehicle. Specifically, we applied $\rho_1 = 20$, $\rho_2 = 25$, $v_1 = 1.2$, and $v_2 = 0.9$ in the simulations. Example routes for 30 targets produced by three different algorithms for both symmetric and asymmetric cases are
presented in Figure 2. In the examples, the heuristics produced the optimal and near optimal solutions within much shorter computation times.

To ascertain the performance, LP relaxation solutions were used to find a posteriori bounds. To solve the LP relaxation, we used a multi-commodity flow formulation on CPLEX [31]. We compared the solutions for the same instances with an LP rounding method, which solves the LP relaxation and assigns each target to the one has the maximum value of the partitioning decision variable. To solve the routing problem within the partition, LKH [32] was used for both Algorithm 2 and Algorithm 3. The results of the computational evaluation is presented in Figure 3 and Tables 2 and 3.

(a) The optimal routes of a TDHTSP: The maximum tour cost was 4950, and the computation time was 4353.1 s.

(b) The optimal routes of a TDHATSP: The maximum tour cost was 5398, and the computation time was 7086.9 s.

(c) The primal-dual routes of a TDHTSP: The maximum tour cost was 4950, and the computation time was 1.96 s.

(d) The primal-dual routes of a TDHATSP: The maximum tour cost was 5416, and the computation time was 5.2 s.

Figure 2. Cont.
The LP rounding routes of a TDHTSP: The maximum tour cost was 6698, and the computation time was 43.22 s.

The LP rounding routes of a TDHATSP: The maximum tour cost was 6975, and the computation time was 19.37 s.

Figure 2. Routes generated by three different algorithms for instances of the cost-symmetric problem (left) and cost-asymmetric problem (right) with 30 targets in a 2000 × 2000 space.

Discussion of Implementation Results

The average and worst a posteriori bounds of each problem size for both cost-symmetric and cost-asymmetric cases are presented in Figure 3. A posteriori bound was calculated by \( \frac{\text{Cost}_{A}(I)}{\text{Cost}_{LP}(I)} \), where \( \text{Cost}_{A}(I) \) represents the cost of the solution obtained through algorithm A of instance I and \( \text{Cost}_{LP}(I) \) represents LP relaxation cost for instance I. Despite the fact that LP rounding usually produced better quality solutions for the min-sum MDHTSP and min-sum MDHATSP [33], the proposed heuristic produced better posteriori bounds for all problem sizes with the min-max objective. For both the symmetric and asymmetric problems, the proposed methods had average posteriori bounds less than 1.4, while LP rounding had bounds greater than 1.6. The worst a posteriori bounds of LP rounding were larger than two for all sizes, while those for the proposed heuristics were less than two.

The average and worst computation times of cost-symmetric and cost-asymmetric cases are summarized in Tables 2 and 3, respectively. In both tables, there exists a huge gap in the computation time between the two algorithms, especially for large-sized problems. For example, the LP rounding method required average of 39,891.46 s, i.e., longer than 11 h, while the proposed heuristic takes approximately 20 s to solve the cost-symmetric problems of 80 targets. As presented in blue and red in Tables 2 and 3, the average computation time was decreased to 0.0005% and 0.001% compare to the LP rounding method for the symmetric and asymmetric cases of 80 targets, respectively, while maintaining better average solution qualities. Therefore, the proposed heuristics are better suitable for applications that require good approximate solutions within a short period of time to increase profits.

Table 2. The computation time in seconds for the TDHTSP.

| No. of Jobs | Average LP Rounding | Average Primal-Dual | Worst LP Rounding | Worst Primal-Dual |
|-------------|---------------------|---------------------|------------------|------------------|
| 20          | 2.11                | 0.81                | 2.91             | 1.61             |
| 30          | 31.13               | 1.79                | 41.23            | 2.35             |
| 40          | 429.61              | 3.30                | 867.66           | 5.26             |
| 60          | 5623.48             | 8.62                | 8470.95          | 11.72            |
| 80          | 39,891.46           | 19.65               | 63,239.30        | 31.02            |
Table 3. The computation time in seconds for the TDHATSP.

| No. of Jobs | Average | Worst |
|-------------|---------|-------|
|             | LP Rounding | Primal-Dual | LP Rounding | Primal-Dual |
| 20          | 1.59     | 1.23   | 2.38       | 3.28       |
| 30          | 17.70    | 4.80   | 22.11      | 12.31      |
| 40          | 237.06   | 4.86   | 345.81     | 12.79      |
| 60          | 3634.76  | 16.00  | 10006.30   | 46.08      |
| 80          | 30,329.19| 36.21  | 41,330.20  | 99.87      |

Figure 3. Posteriori bounds of 50 instances for each problem size.

6. Conclusions

In this research, we proposed heuristics based on a primal-dual technique for the job allocation and routing of a two-UV system with structural heterogeneity while minimizing the last job completion time. The formulations of MIP, LP, and dual problems were presented. Using the dual problems, the heuristics were designed to iteratively run primal-dual algorithms by varying the $W_k$ values, which implies the priority of $k$th vehicle. Through implementation, the effectiveness of our proposed algorithms has been shown.

Our heuristics have limitations for guarantees of worst solution quality, limited number of vehicles, and limited heterogeneity. However, we believe that our heuristics are a good original idea that can solve the min-max MDHTSP in a time-efficient manner while considering additional constraints that increase the complexity of the problem. As future work, we aim to solve the mim-max MDHTSP and min-max MDHATSP while considering both structural and functional heterogeneity with lighter computational loads.

Author Contributions: Conceptualization, J.B. and W.C.; Methodology, J.B.; Validation, J.B.; Writing-Review & Editing, J.B. and W.C; Funding Acquisition, W.C.

Funding: This work was supported by the Industrial Convergence Core Technology Development Program (No. 10063172) funded by MOTIE, Korea, the NRF, MSIP(NRF-2017R1A2A1A17069329) and the Agriculture, Food and Rural Affairs Research Center Support Program (Project No. 714002-07), Ministry of Agriculture, Food and Rural Affairs.

Conflicts of Interest: The authors declare no conflict of interest.
References

1. Reinecke, M.; Prinsloo, T. The influence of drone monitoring on crop health and harvest size. In Proceedings of the 2017 1st International Conference on Next Generation Computing Applications (NextComp), Moka, Mauritius, 19–21 July 2017; pp. 5–10. [CrossRef]

2. Malveaux, C.; Hall, S.G.; Price, R. Using Drones in Agriculture: Unmanned Aerial Systems for Agricultural Remote Sensing Applications. *Am. Soc. Agric. Biol. Eng.* 2014, 1. [CrossRef]

3. Kanistras, K.; Martins, G.; Rutherford, M.J.; Valavanis, K.P. A survey of unmanned aerial vehicles (UAVs) for traffic monitoring. In Proceedings of the 2013 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 28–31 May 2013; pp. 221–234. [CrossRef]

4. Yuan, C.; Zhang, Y.; Liu, Z. A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. *Can. J. For. Res.* 2015, 45, 783–792. [CrossRef]

5. Casbeer, D.W.; Kingston, D.B.; Beard, R.W.; McLain, T.W. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *Int. J. Syst. Sci.* 2006, 37, 351–360. [CrossRef]

6. Mendonça, R.; Marques, M.M.; Marques, F.; Lourenço, A.; Pinto, E.; Santana, P.; Coito, F.; Lobo, V.; Barata, J. A cooperative multi-robot team for the surveillance of shipwreck survivors at sea. In Proceedings of the OCEANS 2016 MTS/IEEE Monterey, Monterey, CA, USA, 19–23 September 2016; pp. 1–6. [CrossRef]

7. Jia, Z.; Yu, J.; Ai, X.; Xu, X.; Yang, D. Cooperative multiple task assignment problem with stochastic velocities and time windows for heterogeneous unmanned aerial vehicles using a genetic algorithm. *Aerosp. Sci. Technol.* 2018, 76, 112–125. [CrossRef]

8. Calhoun, G.; Ruff, H.; Behymer, K.; Frost, E. Human-autonomy teaming interface design considerations for multi-unmanned vehicle control. *Theor. Issues Ergon. Sci.* 2018, 19, 321–352. [CrossRef]

9. Murphy, R.R.; Kravitz, J.; Stover, S.L.; Shoureshi, R. Mobile robots in mine rescue and recovery. *IEEE Robot. Autom. Mag.* 2009, 16, 91–103. [CrossRef]

10. Middleton, W.; Miller, G.; Pollman, A. Architecture models for coordination of unmanned air and ground vehicles conducting humanitarian assistance and disaster relief. In Proceedings of the Conference on Systems Engineering Research, Charlottesville, VA, USA, 8–9 May 2018.

11. Kulich, M.; Kubalík, J.; Preucil, L. An Integrated Approach to Goal Selection in Mobile Robot Exploration. *Sensors* 2019, 19, 1400. [CrossRef] [PubMed]

12. Jungyun Bae, W.C. A Heuristic for Path Planning of Multiple Heterogeneous Automated Guided Vehicles. *Int. J. Precis. Eng. Manuf.* 2018, 19, 1765–1771.

13. Vazirani, V.V. *Approximation Algorithms*; Springer: Berlin/Heidelberg, Germany, 2004.

14. De Oliveira, G.C.; de Carvalho, K.B.; Brandão, A.S. A Hybrid Path-Planning Strategy for Mobile Robots with Limited Sensor Capabilities. *Sensors* 2019, 19, 1049. [CrossRef]

15. Tang, J.; Yang, W.; Zhu, L.; Wang, D.; Feng, X. An Adaptive Clustering Approach Based on Minimum Travel Route Planning for Wireless Sensor Networks with a Mobile Sink. *Sensors* 2017, 17, 964. [CrossRef]

16. Dahan, F.; El Hindi, K.; Mathkour, H.; AlSalman, H. Dynamic Flying Ant Colony Optimization (DFACO) for Solving the Traveling Salesman Problem. *Sensors* 2019, 19, 1837. [CrossRef]

17. Venkatachalam, S.; Sundar, K.; Rathinam, S. A Two-Stage Approach for Routing Multiple Unmanned Aerial Vehicles with Stochastic Fuel Consumption. *Sensors* 2018, 18, 3756. [CrossRef]

18. Bae, J.; Rathinam, S. A primal-dual approximation algorithm for a two depot heterogeneous traveling salesman problem. *Optim. Lett.* 2016, 10, 1269–1285. [CrossRef]

19. Franceschelli, M.; Rosa, D.; Setzru, C.; Bullo, F. Gossip algorithms for heterogeneous multi-vehicle routing problems. *Nonlinear Anal. Hybrid Syst.* 2013, 10, 156–174. [CrossRef]

20. Prasad, A.; Sundaram, S.; Choi, H. Min-Max Tours for Task Allocation to Heterogeneous Agents. In Proceedings of the 2018 IEEE Conference on Decision and Control (CDC), Miami Beach, FL, USA, 17–19 December 2018; pp. 1706–1711. [CrossRef]

21. Stodola, P. Using Metaheuristics on the Multi-Depot Vehicle Routing Problem with Modified Optimization Criterion. *Algorithms* 2018, 11, 74. [CrossRef]

22. Wang, X.; Golden, B.; Wasil, E. The min-max multi-depot vehicle routing problem: Heuristics and computational results. *J. Oper. Res. Soc.* 2015, 66, 1430–1441. [CrossRef]
23. Sundar, K.; Rathinam, S. Algorithms for Heterogeneous, Multiple Depot, Multiple Unmanned Vehicle Path Planning Problems. *J. Intell. Robot. Syst.* **2017**, *88*, 513–526. [CrossRef]

24. Cho, D.H.; Jang, D.S.; Choi, H.L. Sampling-Based Tour Generation of Arbitrarily Oriented Dubins Sensor Platforms. *J. Aerosp. Inf. Syst.* **2019**, *16*, 168–186. [CrossRef]

25. Azadeh, A.; Farrokhi-Asl, H. The close–open mixed multi depot vehicle routing problem considering internal and external fleet of vehicles. *Transp. Lett.* **2019**, *11*, 78–92. [CrossRef]

26. Goemans, M.X.; Williamson, D.P. The Primal-dual Method for Approximation Algorithms and Its Application to Network Design Problems. In *Approximation Algorithms for NP-Hard Problems*; Hochbaum, D.S., Ed.; PWS Publishing Co.: Boston, MA, USA, 1997; pp. 144–191.

27. Helsgaun, K. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* **2000**, *126*, 106–130. [CrossRef]

28. Reeds, J.; Shepp, L. Optimal paths for a car that goes both forwards and backwards. *Pac. J. Math.* **1990**, *145*, 367–393. [CrossRef]

29. Dubins, L.E. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *Am. J. Math.* **1957**, *79*, 497–516. [CrossRef]

30. Sussmann, H.J.; Tang, G. Shortest Paths for the Reeds-Shepp Car: A Worked out Example of the Use of Geometric Techniques in Nonlinear Optimal Control; Rutgers Univ., Tech. Rep.; SYNCON: Piscataway, NJ, USA, 1991.

31. IBM ILOG CPLEX Optimization Studio. 2017. Available online: https://www.ibm.com/us-en/marketplace/ibm-ilog-cplex (accessed on 8 April 2019).

32. LKH-2.0.9. 2009. Available online: http://www.akira.ruc.dk/~keld/research/LKH/ (accessed on 8 April 2019).

33. Bae, J. Algorithms for Multiple Vehicle Routing Problems. Ph.D. Dissertation, Texas A&M University, College Station, TX, USA, 2014.