Decentralized Multiple Mobile Depots Route Planning for Replenishing Persistent Surveillance Robots

Yifan Ding*, Wenhao Luo*, Katia Sycara*

***Abstract*— Persistent surveillance of a target space using multiple robots has numerous applications. The continuous operation in these applications is challenged by limited onboard battery capacity of the persistent robots. We consider the problem for replenishing persistent robots using mobile depots, where mobile depots collectively compute a set of tours to drop off batteries for replenishing all persistent robots with the minimum total cost. Compared to other works, persistent robots are not required to detour for recharging or battery swapping. We formulate this problem as generalized multiple depots traveling salesman problem (GMDTSP) on a complete graph. An efficient centralized heuristic-based algorithm Multiple Depots Random Select (MDRS) is proposed, which has been proved to have an analytical constant upper bound in the worst case scenario. To provide scalability and robustness, a fully decentralized asynchronous MDRS (dec-MDRS) is proposed, with the analysis of its correctness and efficiency. We also propose a post-processing heuristic (MDRS-IM) to improve the solution quality. We demonstrate the efficiency and effectiveness of our algorithm via benchmark on multiple instances from TSPLIB and GTSPLIB. The simulation results show that the complexity of dec-MDRS grows linearly as the number of robots increases. The simulation also shows that MDRS and MDRS-IM perform significantly faster than the state of the art heuristic solver LKH with only a loss of about 10% of solution quality.**

I. Introduction

Persistent surveillance of a target space using multiple robots has numerous applications such as geographical surveys, air quality monitoring, and security monitoring [1], [2], [3]. Limited onboard battery capacity is one of the challenges for persistent robots to execute persistent tasks continuously. To address the long term operation for persistent surveillance tasks, route planning for periodic recharging or battery swapping becomes one of the popular research questions.

There is an extensive literature on replenishing persistent robots by placing static depots, where the persistent robots need to detour for replenishment [4], [5], [6]. These literature discuss the optimal quantity and placement for the static depots which lead to a minimum cost for the persistent robots to recharge. However, persistent robots need to be removed from the tasks for replenishment, which becomes a challenge for safety-critical missions. Moreover, due to the dynamics for the environment, the optimal placement always changes to time and is relevant to the persistent missions.

In this paper, multiple mobile depots are deployed to replenish persistent robots so that they are not required to detour for replenishment. Fig. 1 shows an example for our formulation, where there are two types of robots, persistent robots and mobile depots. Persistent robots, with limited onboard battery capacity, move on a pre-planned fixed route to monitor their prescribed areas. Before persistent robots run out of battery, they determine a set of potential preferred battery swapping locations. Mobile depots, which assumed to carry unlimited batteries, move among persistent robots to drop off batteries at some battery swapping locations. To minimize detour for persistent robots, we assume all potential preferred battery swapping locations are on its pre-planned route. Once persistent robots travel to the places where the battery is dropped off, they swap the battery to replenish itself. The persistent robots only accept to swap battery at one of their preferred locations. The problem we need to solve is to find tours for the mobile depots to replenish all persistent robots within their preferred battery swapping locations with the minimum total cost. We formulate this problem as Generalized Multiple Depots Traveling Salesman Problem (GMDTSP), which turns out to be an NP-hard problem. The challenge for this problem is 1) how to assign persistent robots to different mobile depots; 2) what is the optimal order to visit the persistent robots; 3) what is the optimal location to drop off batteries given each persistent robot has multiple preferred locations.

This paper presents three main contributions. First, a heuristic-based algorithm Multiple Depots Random Select (MDRS) is proposed to efficiently solve the route planning

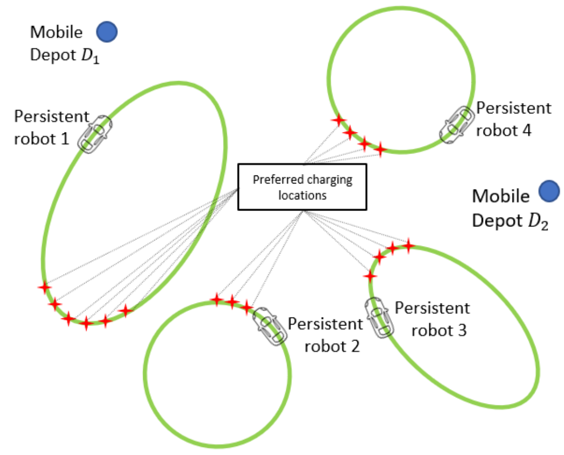

Fig. 1. Example for the multiple mobile depots route planning problem, with 2 mobile depots and 4 persistent robots. The red asterisks are the preferred battery swapping locations. The green paths are the pre-planned routes for persistent robots.

*The authors are with the Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Email: {yifand1, wenhaol}@andrew.cmu.edu, katia@cs.cmu.edu

This work was funded by Darpa award HR0011-18-2-0051.

problem for multiple mobile depots for replenishing persistent surveillance robots. The problem is modeled as a GMDTSP on a complete graph. The MDRS has proved to have an analytical constant upper bound in the worst case scenario. Second, a fully decentralized asynchronous MDRS (dec-MDRS) is proposed to provide scalability and robustness. Simulations show that the computation time and the number of messages of dec-MDRS grows linearly as the number of robots increases. Third, a post-processing heuristic (MDRS-IM) is proposed to improve the solution quality further. The simulation results show that MDRS and MDRS-IM perform significantly faster than the state of the art heuristic solver LKH [7] with only a loss of about 10% of solution quality.

II. Related Work

Recharging or swapping batteries for persistent robots using mobile depots has different formulations in the literature. Different than route planning for multiple mobile depots, [8], [9] formulate the problem as a single mobile depot planning a route for recharge the UAVs. The single mobile depot problem can be solved by the transformation method, mixed integer programming (MIP) or heuristic-based algorithm. In [10], [11], [12], a battery swapping system has been modeled, with the assumption that the swap can happen instantly without the charging duration. However, the battery swapping needs to have fully charged battery in stock which may significantly increase the operation cost.

A recent work [13] has a similar formulation, where a heuristic-based solution performs efficiently, but the effectiveness is only shown based on empirical simulation results instead of an analytical analysis. Also, the algorithm did not admit the periodic structure. Similar to this work, [14] discretize the state space for periodic recharges and cast this problem as a generalized traveling salesman problem (GTSP) on a partitioned directed acyclic graph. With the transformation, the state of the art LKH solver [7] is used to solve the GTSP. However, in our work, the problem is formulated as a GMDTSP on a complete graph. Moreover, the work [14] use the modified noon-bean transformation [15] to transform the problem to a traditional TSP problem, which increases the size of the vertex set.

We formulate the problem as a GMDTSP, where an exact MIP formulation exists [16]. However, the computation time is not acceptable for the real-time application. An efficient and effective heuristic-based algorithm is needed for solving this problem.

III. Problem Formulation

Consider n persistent ground robots moving in $\mathbb{R}^2$, following their pre-planned routes to continuously monitor their prescribed areas. To address the long term operation for the persistent surveillance tasks, the persistent robots need to swap battery at some locations before running out of battery due to the limited onboard battery capacity. Assume that each persistent robots $i \in \{1, 2, \cdots, n\}$ can figure out a set of disjoint potential preferred battery swapping locations C_i based on their knowledge, which consists of a set of discretized vertices on its pre-planned route. The persistent robot i only accept to swap battery at one of its own preferred locations in C_i since no detour is allowed. Consider k holonomic mobile depots $D = \cup d_i,\ \forall i \in \{1, 2, \cdots, k\}$ carrying unlimited batteries, where d_i is the position in $\mathcal{R}^2$ of the depots initially. The mobile depots collectively plan a set of tours to replenish all persistent robots with the minimum total cost. The mobile depots drop off a battery at each stop of the tour so that the persistent robot can swap battery once it travels to this preferred location.

Fig. 1 shows an example for the multiple mobile depots route planning problem, with two mobile depots ($k = 2$) and four persistent robots ($n = 4$). The red asterisks are the preferred battery swapping locations, i.e., for a valid battery swap, the mobile depots should only drop off batteries at these red asterisks.

The multiple mobile depots route planning problem can be formulated on a complete undirected graph $G = (V, E, c)$ with vertex set $V = C \cup D$. The cost of an edge $e(p, q) \in E$ is assigned to be the Euclidean distance between vertices $p, q \in V$. The mobile depots collectively compute a set of at most k tours $TOUR_i, i \in \{1, 2, \cdots, k\}$. The tour is defined as a simple circle on G where each mobile depot ends at its starting location, i.e., $TOUR_i^{\text{end}} = TOUR_i^{\text{start}}$. At least one vertex from each vertex set $C_i, \forall i \in \{1, 2, \cdots, n\}$ is visited by at least one mobile depot. The formal formulation for this problem then becomes,

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{k} c(TOUR_i), \\
\text{subject to} \quad & TOUR_i^{\text{end}} = TOUR_i^{\text{start}}, \quad i = 1, \cdots, k, \\
& |\cup_{i=1}^{k} TOUR_i \cap C_j| \geq 1, \quad j = 1, \cdots, n.
\end{aligned}$$

Note that this problem formulation falls into the category of GMDTSP. When the vertex set $C_i, \forall i \in \{1, 2, \cdots, n\}$ only contains a single vertex, then this problem degenerate to a MDTSP. When $k = 1$, then this problem degenerate to a GTSP.

Remark 1. *The graph G is a symmetric graph, where $c(p, q) = c(q, p)$ for all vertices $p, q \in V$. For non-holonomic mobile depot, the edge cost may not be symmetric due to the dynamic constraints. In our problem formulation, all mobile depots are holonomic.*

Remark 2. *The complete graph G satisfies the triangle inequality since the edge cost is defined by Euclidean distance between vertices. Based on the problem formulation, it is valid that multiple mobile depots drop off batteries to the same persistent robot. However, in fact, to minimized to total cost for all k tours, each persistent robot will only be served by one and only one mobile depot due to the triangular inequality [17].*

Remark 3. *The current cost only reflect the Euclidean distance between vertices, some other factors such as travel time and battery swapping time can also be associated with the cost as long as the new cost satisfy the triangle inequality.*

IV. Multiple Depot Random Select and Analysis

We present a centralized heuristic-based Multiple Depot Random Select (MDRS) in Algorithm 1. MDRS first reduces the GMDTSP to a MDTSP by randomly select a vertex in each vertex set, then uses a modified MST-based algorithm [13] for the resulting MDTSP to obtain an approximate solution. In this algorithm, the selected vertices can represent all vertices to reduce complexity. The underlying nature is that every vertex in the vertex set $C_i, \forall i \in \{1, 2, \cdots, n\}$ correlates with others, given every vertex are the preferred battery swapping locations for the same persistent robot.

In detail, Algorithm 1 first randomly select a vertex from each disjoint vertex set $C_i, \forall i \in \{1, 2, \cdots, n\}$ to form a subset $P \subseteq C$ on line 1 to 2, and $|P| = n$. Inspired by the well-known minimum spanning tree (MST) heuristic in TSP and multiple TSP (mTSP), line 3 build a weighted undirected graph on vertex set P, D and a dummy node v_{dummy}, where line 10 to 14 describes how we define the edges with cost. Prim's algorithm is used to find the minimum spanning tree (MST) on the newly built graph. The MST has a total of $|P| + |D| + 1$ nodes if we includes the dummy node v_{dummy}. For all k depot nodes $d_i \in D$, line 6 break the MST into k subtrees T_i rooted at d_i. A depth first search is executed on each subtree T_i, starting from the corresponding root d_i. The visitation sequence S is recorded. Line 8 bypass the duplicated vertices in sequence S, and $TOUR_i$ is returned. Using the well-known inequality scaling technique [18], MDRS can be proved to have an analytical constant upper bound in the worst case scenario.

Theorem 1. *MDRS(G) is the route cost constructed by MDRS on G, and OPT(G) is the optimal route cost on G. Then,*

$$\frac{\textit{MDRS}(G)}{\textit{OPT}(G)} \leq 2(1 + 2d/\rho)$$

where $d := \max_{i=1}^{n}\{\max_{p,q \in C_i} dist(p, q)\}$ is the maximum inter-vertex set distance, and $\rho := \min dist(p, q)$, where $p \in C_i, q \in C_j, i \neq j$ is the minimum intra-vertex distance.

Proof: Given the problem formulation, $G = (V, E, c)$ is a complete undirected graph. $G' = (P, E', c')$, built by Algorithm 1, is an undirected subgraph of G. Note that the GMDTSP problem on G is degenerate to a TSP on G'.

Since G' is one of the selected subgraph for G, without loss generality, we have MDRS(G)$\leq$ MDRS(G'). Assume OPT(G') is the cost of the optimal route on G', one can prove the minimum spanning tree heuristics with bypassing has a constant upper bound of 2 [19]. We then have MDRS(G') is no greater than $2 \cdot$ OPT(G'). Therefore, combine these two inequalities, we have

$$\text{MDRS}(G) \leq 2 \cdot \text{OPT}(G') \tag{1}$$

The obtained optimal TSP sequence on G' is some permutation on vertex set P, and define this optimal sequence as S. Define S' as another permutation on P, where S' has the same order as the optimal GMDTSP vertex set visitation sequence on the original graph G. We can imply that OPT$(G') = cost(S) \leq cost(S')$. Since S' is the optimal

Algorithm 1: Multiple depot random select (MDRS) heuristics

```
Inputs   : $C := \cup C_i, \forall i \in \{1, 2, \cdots, n\}, D$
Outputs : $k$ tours $TOUR_i$ for $i = 1, \cdots, k$
1  Randomly select a vertex $v_i \in C_i, \forall i \in \{1, 2, \cdots, n\}$
2  $P := \cup v_i, \forall i \in \{1, 2, \cdots, n\}$
3  $G'$ = buildGraph($P$, $D$)
4  $M$ = mst($G'$)
5  for $i = 1 : k$ do
6      Break $M$ into subtree $T_i$ rooted at $d_i \in D$
7      $S$ = traverse($T_i$) using depth first search
8      $TOUR_i$ = bypass($S$) by eliminating previously
         occurred vertices
9  end
   Function buildGraph($P, D$)
10     Set $V = P \cup D \cup v_{dummy}$
11     Add $e(v_{dummy}, i)$ to $E$ with $c = 0$, $\forall i \in D$
12     Add $e(i, j)$ to $E$ with $c = dist(i, j)$, $\forall i, j \in P$
13     Add $e(i, j)$ to $E$ with $c = dist(i, j)$, $\forall i \in D, j \in P$
14     return undirected graph $G = (V, E, c)$
```

vertex set visitation sequence on set P, using the triangular inequality, we can have $cost(S') \leq$ OPT$(G) + 2n \cdot d$. Based on the definition on ρ, we have OPT$(G) \geq n \cdot \rho$. Combine these inequalities, we have

$$\text{OPT}(G') \leq (1 + 2d/\rho)\text{OPT}(G) \tag{2}$$

The theorem is proved by combining the inequality (1) and (2). □

Theorem 2. *MDRS returns a feasible set of tours satisfying the problem formulation. In other words, edges $e(v_{dummy}, i)$ for all $i \in D$ are edges for the MST on G' rooted at the dummy node v_{dummy}. Mobile depot i has a sub-tree $T_i, i \in \{1, 2, \cdots, k\}$, where $|T_i \cap T_j| = \varnothing$, and $|T_i \cup T_j| = C$, where $i, j \in \{1, 2, \cdots, k\}, i \neq j$.*

Proof: If we apply the Prim's greedy algorithm, then all mobile depots will be connected to the dummy node v_{dummy} since the edge cost between them is the smallest, zero. With the connection between the dummy node and all mobile depots, any vertex in C will not connect to dummy node since the edge cost between them is infinite. This implies $|T_i \cup T_j| = C$. Also, the sub-tree rooted at the mobile depots will not intersect with each other. Otherwise, a circle will be formed which contradict with the concept of tree. This implies $|T_i \cap T_j| = \varnothing$. □

V. Decentralized MDRS and Analysis

In this section, we introduce a decentralized framework to implement our approach which provides scalability and robustness. Since persistent tasks can evolve, the decentralized framework could be more robust to individual failures or time delays. The goal for the fully decentralized MDRS (dec-MDRS) heuristic is to obtain the solution by asynchronously passing messages among all persistent robots and mobile depots. In Section IV, the centralized MDRS plans the paths for each mobile depots, while in the dec-MDRS, both persistent robots and the mobile depots are involved in the planning framework. The persistent robots are included in the planning process to enlarge the communication range and increase the computational speed.

The dec-MDRS works as follows. 1) Each persistent robot will randomly select a battery swapping location from its preferred locations set. 2) GHS algorithm [20], a well known asynchronous distributed algorithm, will be used to construct minimum spanning tree. The constructed graph G' is the same as in the centralized version. Each robot knows the adjacency list with weights for itself. Assume the graph G' has n nodes and m edges, GHS algorithm runs in $O(n \log_2 n)$ time and using at most $O(m + n \log_2 n)$ messages. The output for this step is a minimum spanning tree, which is represented as the adjacency list of vertices adj_{MST} stored in each robot. 3) With the MST built, the last step is to traverse the tree and bypass the repeated vertices to form a tour. We present the last step in Algorithm 2.

Algorithm 2 describes the traversal and bypassing for robot r_i, where the robot can be both mobile depots and persistent robots. Before this algorithm starts, r_i knows its adjacency list of vertices adj_{MST}, which is a list of its neighbor vertices on the built MST. Note that adj_{MST} can be unsorted. Also, if the robot is a mobile depot, then it knows the Boolean $first$ is true, meaning mobile depots are the robots that kick off the algorithm. The output is the next vertex $next$ on the constructed $TOUR$. Each robot r_i has three internal variables to keep track its own status, $count$, $started$ and $visited$. $count$ is an integer variable which points the position of the adj_{MST}, which is initialized to be 1, meaning the first element in adj_{MST} has been pointed. Boolean $visited$ means whether it has been visited before, which is initialized to be $false$. Another Boolean $started$ means whether this robot has been woken up, which is initialized to be $false$. The messages has the same protocol, where $msg = \langle type, src, [optional]other \rangle$. Robots asynchronously send and receive three message types to each other with the known protocol to construct the tour. When a robot is been visited for the first time, it marks itself as $visited$. Line 13 to 17 find the its parent in adj_{MST} and move it to the last position in the list. Once the counter hit the end of adj_{MST}, the robot traces back to its parent as in the depth first search. The robot then increases the counter and do the same traversal to the next vertex in adj_{MST}. If the robot is visited before, this means the robot needs to be bypassed. This is achieved by sending messages to the two related bypass neighbors. Assume that (r_j, r_i) and (r_i, r_k) is edges on the built MST, and robot r_i is bypassed. r_i will send message to r_j telling r_k is next vertex $next$ on the constructed tour, and send another similar message to r_k with the vertex r_j. The algorithm is terminated once the counter $count$ points out of adj_{MST}.

Theorem 3. *Algorithm 2 eventually terminates and constructs the correct set of tours.*

Proof: Algorithm 2 traverses the undirected edge at most twice in adj_{MST}, which means finite number of edges will eventually leads to termination. In other words, since the counter $count$ always increases while the length of adj_{MST} is unchanged, the counter will eventually meet the termination condition.

In terms of correctness, Algorithm 2 starts on the set of mobile depots. Once the robot is been visited for the first time, the robot marks itself as visited and remains the status. The robot will also mark its parent during the first visit. This ensures the robot can trace back once all other neighbors has been visited. All later visitation on those robots who has already been visited will be bypassed. ☐

Algorithm 2: Distributed Traverse and Bypass Algorithm, for robot r_i

```
Inputs   : Adjacency vertex list $adj_{MST}$, Boolean $first$
Outputs  : Next vertex $next$ on $TOUR_i$
1  Initialize $count = 1, visited = false, started = false$
   When receiving no message:
2      if $started = false$ then
3          $started = true$
4          if $first = true$ then proceed(-1)
5      end
   When receiving message $\langle A, src \rangle$:
6      if $count > |adj_{MST}|$ then Terminate
7      if $visited = false$ then proceed($src$)
8      else bypass($src$)
   When receiving message $\langle B, src, vertex \rangle$:
9      $next = vertex$
   When receiving message $\langle C, src, vertex \rangle$:
10     if $count > |adj_{MST}|$ then Terminate
11     if $visited = false$ then proceed($src$)
12     else bypass($vertex$)
   Function proceed($src$)
13     $visited = true$
14     if $src \neq -1$ then
15         Delete $adj_{MST}(adj_{MST} == src)$
16         Insert $src$ to the end of $adj_{MST}$
17     end
18     $count = count + 1$
19     $next = adj_{MST}(count)$
20     sendmsg($\langle A, r_i \rangle$, $adj_{MST}(count)$)
   Function bypass($r_j$)
21     $r_k = adj_{MST}(count)$
22     sendmsg($\langle B, r_i, r_k \rangle$, $r_j$)
23     sendmsg($\langle C, r_i, r_j \rangle$, $r_k$)
24     $count = count + 1$
   Function sendmsg($msg$, $dst$)
25     Send message $msg$ to destination $dst$
```

VI. Improvement Heuristics on Tours

In this section, a centralized improvement heuristics (MDRS-IM) is proposed to increase the solution quality. MDRS introduced in Section IV randomly select a vertex from each of the vertex set to represent the whole set. MDRS-

IM tries to compensate for the "loss" of options that occurred during sampling by fixing the order of the visits but re-allowing to consider the other possible charging locations.

MDRS generates a set of tours for each mobile depots, where each mobile depot knows a set of persistent robots need to be served with a visitation sequence. Define a map $\mathcal{M} := \mathbb{R}^2 \rightarrow \mathbb{N}$, which maps the randomly selected vertex $v_i \in C_i$ to the corresponding vertex set index $i \in \{1, 2, \cdots, n\}$. Given a tour T, define $T(j)$ as the j^{th} vertex in T. Define two tour T_1 and T_2 are equal if and only if 1) $|T_1| = |T_2|$, and 2) $T_1(j) = T_2(j)$ for all $j = \{1, 2, \cdots, |T_1|\}$. Otherwise, $T_1 \neq T_2$. Define $TSP(T)$ as a tour obtained on tour T using TSP algorithm.

Algorithm 3 describes the MDRS-IM procedure to improve the solution quality. The input T_{d_i} is the tour generated by MDRS for mobile depot d_i. The output is the improved tour T_{IM} with a better quality for this mobile depot. The algorithm build a graph $G_{IM} = (V_{IM}, E_{IM}, c_{IM})$ from Line 5 to Line 20 based on the input group visitation sequence. Vertices in neighbor groups are fully connected, and all vertices in the first and last group are connected to the dummy vertex s and t. The shortest path search such as A* return the optimal tour given the current group visit sequence. Line 4 is for continuous improvement based on the previous post-processing result until the improved heuristics get the local minimum.

Algorithm 3: Improvement Heuristics on MDRS (MDRS-IM) for mobile depot d_i

```
Inputs   : The original tour $T_{d_i}$ from MDRS
Outputs : The improved tour $T_{IM}$
1  Initialize $G_{IM} = (V_{IM}, E_{IM}, c_{IM})$ with $V_{IM} = V$,
2  $T_{IM} = \varnothing$
3  $L := |T_{d_i}|$
4  while $T_{IM} == \varnothing$ OR $TSP(T_{IM}) \neq T_{IM}$ do
5      for $j = 0, 1, \cdots, L-1$ do
6          if $j = 0$ then
7              Insert dummy node $s$, $V_{IM} = V_{IM} \cup s$
8              Connect $E_{IM}(s, v)$, $\forall v \in C_{\mathcal{M}(T(1))}$
9              Set $c_{IM}(s, v) = 0$, $\forall v \in C_{\mathcal{M}(T(1))}$
10         end
11         else if $j = L-1$ then
12             Insert dummy node $t$, $V_{IM} = V_{IM} \cup t$
13             Connect $E_{IM}(v, t)$, $\forall v \in C_{\mathcal{M}(T(L))}$
14             Set $c_{IM}(v, t) = 0$, $\forall v \in C_{\mathcal{M}(T(L))}$
15         end
16         else
17             Connect $E_{IM}(u, v)$ for all $u \in C_{\mathcal{M}(T(j))}$,
                $v \in C_{\mathcal{M}(T(j+1))}$
18             Set $c_{IM}(u, v) = dist(E_{IM}(u, v))$
19         end
20     end
21     $T_{IM} \leftarrow$ shortest path from dummy node $s$ to $t$
22 end
```

VII. Simulation Results

The proposed algorithms are implemented in C++ with an open source template graph library LEMON[1]. The library provides efficient implementations of common data structures and graph algorithms. Simulation results were computed on a laptop running a 64 bit Ubuntu 16.04 operating system with an Intel® Core™i7-8550U CPU @1.80GHz x 8 and 16GB of RAM.

Fig. 2 illustrates an example of the multiple mobile depots route planning problem. Here we have $k = 2$ mobile depots shown in black diamonds. $n = 9$ persistent robots with their pre-planned paths are shown in green dotted lines, with $|C| = 70$ total potential charging locations shown in red asterisks on the green path. MDRS-IM is used to construct the set of tours, with the solution shown in the blue solid line, where mobile depots end the tour at their initial positions.

Fig. 3 investigates the relation between the solution quality ratio versus d/ρ. The solution quality ratio is defined as the cost obtained by MDRS-IM versus the known optimal cost. The optimal cost is calculated using the MIP optimization formulation proposed by [16]. The MIP is solved by IBM CPLEX®. We run 200 instances in total to compile Fig. 3, which shows the mean, standard deviation and the extremes for various d/ρ. The instances are created as follows. First, select 20 instances in TSPLIB[2]. Then, for each point on the TSP instance, we use 2D Gaussian distribution to generate 20 points to form a corresponding vertex set. Each vertex set is treated as the set of preferred charging locations for a persistent robot. We randomly distribute $k = 2$ depots on the constructed graph. We can obtain various d/ρ by controlling the μ and covariance matrix for the Gaussian distribution. Fig. 3 shows that the mean of cost ratio will gradually increase with the increment of d/ρ. In most cases, the algorithm obtains a cost ratio within 1.2.

Table I shows the time and solution quality comparison between LKH with our MDRS and MDRS-IM on GT-SPLIB[3][21] instances. In GTSPLIB, 'Name' indicates the vertex set number and the total vertex number. For example, instance '99d493' means 99 vertex sets and 493 total vertices. The column 'Opt' in Table I means the optimal tour cost for the GTSP instances, while the 'Value' columns are the tour costs given by different methods. The 'ratio' column is defined by the ratio between 'Value' and 'Opt', where the performance is better if the ratio is closer to 1. The 'Value' and 'Time' for LKH method are directed benchmarked using GLKH version 1.0 from [22]. For simplicity, all instances are route planning for the single mobile depot. Both MDRS and MDRS-IM computes significantly faster than the LKH solver for these very large instances with little solution quality loss.

Fig. 5 shows our proposed dec-MDRS approach has good

[1]Open source library LEMON, `https://lemon.cs.elte.hu/trac/lemon`.

[2]TSPLIB instances collected by Gerhard Reinelt, `https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/`.

[3]GTSP Instances Library (GTSPLIB) collected by Daniel Karapetyan, `http://www.cs.nott.ac.uk/~dxk/gtsp.html`.

TABLE I

TIME AND SOLUTION QUALITY COMPARISON BETWEEN LKH WITH MDRS AND MDRS-IM ON GTSPLIB.

Name	Opt	LKH			MDRS (Ours)			MDRS-IM (Ours)		
		Value	Time [s]	Ratio	Value	Time [s]	Ratio	Value	Time [s]	Ratio
99d493	20023	20023.4	170.5	1.00	26168	0.0	1.31	22504	0.2	1.12
132d657	22498	22498	490.8	1.00	28136	0.0	1.25	25333	0.2	1.13
134gr666	163028	163028	162.3	1.00	204543	0.0	1.25	184782	0.2	1.13
145u724	17272	17272	145.4	1.00	21640	0.0	1.25	19285	0.2	1.12
157rat783	3262	3262.9	764.4	1.00	4172	0.0	1.28	3646	0.2	1.12
200dsj1000	9187884	9187884	794.4	1.00	11954184	0.1	1.30	10310418	0.2	1.12
201pr1002	114311	114311	164.8	1.00	150757	0.1	1.32	129730	0.3	1.13
207si1032	22306	22306	1202.5	1.00	28315	0.1	1.27	25135	0.2	1.13
212u1060	106007	106029.5	2054.9	1.00	136547	0.1	1.29	116941	0.3	1.10
217vm1084	130704	130704	209.0	1.00	169315	0.2	1.30	149974	0.3	1.15

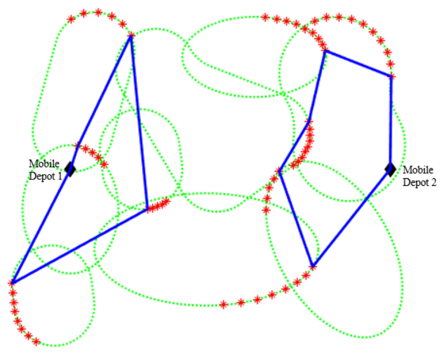

Fig. 2. An illustrative example showing the constructed tours (blue solid line) using MDRS-IM. Black diamonds are mobile depots, green dotted lines are the pre-planned routes for persistent robots. Red asterisks are the preferred battery swapping locations. It can be verified that this path is optimal in terms of minimizing the total costs. It can also be verified that dec-MDRS also produce the same tour set.

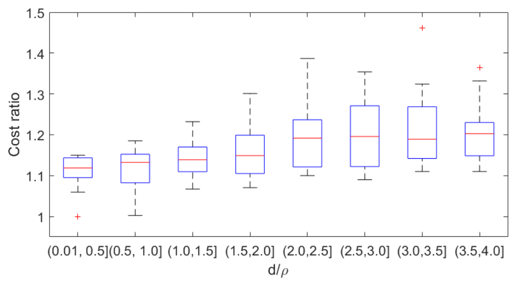

Fig. 3. Relation between the solution quality ratio versus d/ρ.

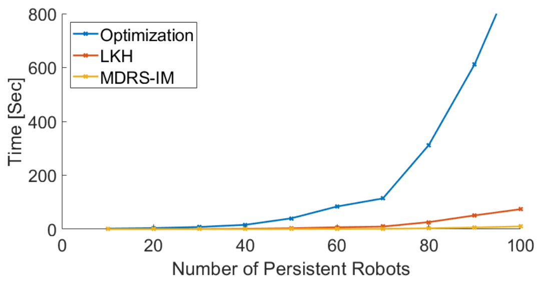

Fig. 4. Comparison of computation time between three methods: optimization, LKH and MDRS-IM.

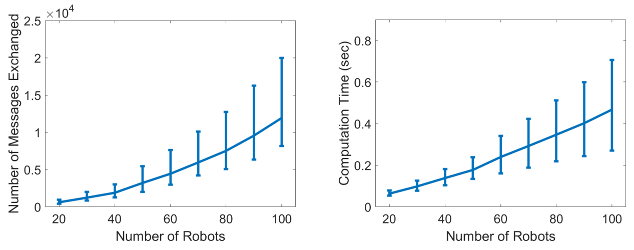

Fig. 5. Results from our proposed dec-MDRS approach. Left figure shows number of messages exchanged to construct tours using dec-MDRS. The error bar shows the maximum and minimum number of messages exchanged. Right figure shows computation time of constructing dec-MDRS.

scalability. As the number of robots increases, the messages exchanged and the compulation time grows linearly. The correctness of dec-MDRS can also be verified in the example in Fig. 2, where the decentralized algorithm return the same set of tours as the centralized algorithm.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we consider the problem for replenishing persistent robots using mobile depots. We formulate this problem as a GMDTSP on a complete graph. An efficient centralized heuristic-based algorithm MDRS is proposed. MDRS captures the nature for GMDTSP, where the vertices in the same vertex set have a close correlation, and thus a single vertex has good representation for the whole vertex set. This nature dramatically reduces the problem complexity and computation time. We also propose a centralized post-processing heuristic (MDRS-IM) to improve the solution quality further. dec-MDRS is then proposed to provide scalability and robustness for the centralized version of the algorithm. In the future, we may corporate time into the problem formulation so that the dynamics of both persistent robots and mobile depots can be considered in route planning. Moreover, several aspects such as sensor noise, unmapped obstacles in stochastic environments need to be analyzed to bridge the gap between the ideal simulation environment and reality.

References

[1] N. Nigam and I. Kroo, "Persistent surveillance using multiple unmanned air vehicles," in *2008 IEEE Aerospace Conference*, pp. 1–14.

[2] E. Stump and N. Michael, "Multi-robot persistent surveillance planning as a vehicle routing problem," in *2011 IEEE International Conference on Automation Science and Engineering*, pp. 569–575.

[3] A. McGonigle, A. Aiuppa, G. Giudice, G. Tamburello, A. Hodson, and S. Gurrieri, "Unmanned aerial vehicle measurements of volcanic carbon dioxide fluxes," *Geophysical research letters*, vol. 35, no. 6, 2008.

[4] J. Kim and J. R. Morrison, "On the concerted design and scheduling of multiple resources for persistent uav operations," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 479–498, 2014.

[5] D. Mitchell, M. Corah, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot long-term persistent coverage with fuel constrained robots," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1093–1099.

[6] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[7] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[8] Z. Jin, T. Shima, and C. J. Schumacher, "Optimal scheduling for refueling multiple autonomous aerial vehicles," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 682–693, 2006.

[9] J. W. Barnes, V. D. Wiley, J. T. Moore, and D. M. Ryer, "Solving the aerial fleet refueling problem using group theoretic tabu search," *Mathematical and computer modelling*, vol. 39, no. 6-8, pp. 617–640, 2004.

[10] K. A. Suzuki, P. Kemper Filho, and J. R. Morrison, "Automatic battery replacement system for uavs: Analysis and design," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 563–586, 2012.

[11] K. A. Swieringa, C. B. Hanson, J. R. Richardson, J. D. White, Z. Hasan, E. Qian, and A. Girard, "Autonomous battery swapping system for small-scale helicopters," in *2010 IEEE International Conference on Robotics and Automation*, pp. 3335–3340.

[12] D. Lee, J. Zhou, and W. T. Lin, "Autonomous battery swapping system for quadcopter," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 118–124.

[13] N. Kamra, T. S. Kumar, and N. Ayanian, "Combinatorial problems in multirobot battery exchange systems," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 852–862, 2018.

[14] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot rendezvous planning for recharging in persistent tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 128–142, 2015.

[15] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR: Information Systems and Operational Research*, vol. 31, no. 1, pp. 39–44, 1993.

[16] K. Sundar and S. Rathinam, "Generalized multiple depot traveling salesmen problempolyhedral study and exact algorithm," *Computers & Operations Research*, vol. 70, pp. 39–55, 2016.

[17] M. Fischetti, J. J. Salazar González, and P. Toth, "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem," *Operations Research*, vol. 45, no. 3, pp. 378–394, 1997.

[18] A. Sadeghi and S. L. Smith, "Heterogeneous task allocation and sequencing via decentralized large neighborhood search," *Unmanned Systems*, vol. 5, no. 02, pp. 79–95, 2017.

[19] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing." in *Robotics: Science and Systems*, vol. 5. Rome, Italy, 2005, pp. 343–350.

[20] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and systems (TOPLAS)*, vol. 5, no. 1, pp. 66–77, 1983.

[21] G. Gutin and D. Karapetyan, "A memetic algorithm for the generalized traveling salesman problem," *Natural Computing*, vol. 9, no. 1, pp. 47–60, 2010.

[22] K. Helsgaun, "Solving the equality generalized traveling salesman problem using the lin–kernighan–helsgaun algorithm," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 269–287, 2015.