Combinatorial Problems in Multi-Robot Battery Exchange Systems

Nitin Kamra, T. K. Satish Kumar, and Nora Ayanian, *Member, IEEE*

***Abstract*—This paper addresses combinatorial problems that arise in multi-robot battery exchange systems. The multi-robot battery exchange system addressed herein is characterized by two types of robots: (a) task robots that provide services at requested locations; and (b) delivery robots that deliver charged batteries to task robots when required. Combinatorial problems arising in these systems involve multiple aspects of resource scheduling and path planning that make them more complex than well-known combinatorial problems studied in operations research. We present several heuristic algorithms for solving these combinatorial problems. Our algorithms are inspired by techniques used in artificial intelligence and the design of approximation algorithms. We demonstrate the performance of our algorithms in simulation and analyze how they scale with increasing size of the multi-robot system.**

Note to Practitioners: This paper studies combinatorial problems arising in multi-robot systems that intend to sustain themselves using mechanisms for battery exchange. The ideas developed in this paper are broadly applicable to resource delivery and scheduling, closed-loop product supply management, and pickup and delivery problems for transportation. Many of the problems studied in this paper are generalizations of the Traveling Salesman Problem (TSP). Simpler variants of these have been studied in operations research with the aim of finding optimal solutions—albeit in exponential time. However, in this paper, we present heuristic algorithms for solving battery exchange problems based on techniques developed in artificial intelligence and the design of approximation algorithms. Our algorithms have polynomial execution times and therefore scale very well with increasing size of the system. More expressive variants of battery exchange problems, with a broader class of constraints, must be addressed in the future. In addition, future research should also emphasize decentralized algorithms in order to encourage parallelism and exploit the computing power available on each individual robot.

***Index Terms*—Energy-aware planning, multi-robot systems, battery exchange, mobile robots, persistent robotics, vehicle routing, path planning, scheduling.**

I. Introduction and Related Work

Multi-robot path planners which obey energy constraints are key components of persistent robotic systems. Limited on-board power is a key challenge for mobile robots on long-duration missions. While there has been an increasing interest in lifelong deployment of robot teams, most of the current literature addressing the issue of limited power requires robots to either return to their docking stations to recharge or spend significant time recharging on-field, both of which can disrupt the execution of long-duration plans.

Recent work in persistent robotics introduces a persistent operation scheduling algorithm that allows Unmanned Aerial Vehicles (UAVs) to return to their docking stations for recharging [1]. Similarly, Derenick *et al.* propose an energy-aware multi-robot system for coverage that alters the team's configuration to allow low-energy robots to return for recharging [2]. These approaches repeatedly remove robots from their assigned tasks and thus cannot be used for accomplishing critical long-duration tasks. Song *et al.* avoid this problem by replacing discharged robots with new ones but at the cost of significantly increasing the number of robots required [3]. In all of these works, travel from and to a charging station can waste a significant amount of energy, reducing the usable lifespan of robot during a cycle. Kannan *et al.* introduce the Autonomous Recharging Problem (ARP) and propose an energy-aware design along with static and mobile charging stations [4]. Mathew *et al.* allow on-the-spot recharging for persistent task UAVs but unrealistically assume instantaneous recharging [5]. Even if this assumption is obviated, the framework, like that of Kannan *et al.*, prohibits a recharging robot from attending to its assigned tasks. Other works present persistent monitoring schemes but ignore battery exhaustion altogether [6].

Several hardware-only solutions for recharging and docking mobile robots have been developed [7]–[10]. However, these methods take the robots out of service while recharging. The idea of distributable energy was suggested by Ngo *et al.* with a policy for energy sharing [11]. While battery sharing among robots can prolong their working durations, it cannot completely avoid the problem of robots eventually having to return to their docking stations for recharge.

Recent works on autonomous battery swapping mechanisms, many of which are specifically tailored for UAVs [12]–[17], can be used to avoid wasted time and energy incurred while robots return to and from charging stations. While this requires a supply of fully charged batteries in preparation for deployment, the cost of keeping additional batteries available is significantly lower than the cost of deploying additional robots that would otherwise be necessary to ensure uninterrupted operations. Autonomous battery swapping methods can be used for changing batteries on the field, effectively eliminating robot downtime at the marginal cost of having to keep a relatively small number of delivery robots on hand.

This paper addresses combinatorial problems that arise in

Authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, 90089, USA. email: {nkamra, ayanian}@usc.edu, tkskwork@gmail.com. This work was supported by Office of Naval Research grant N00014-14-1-0734.

Manuscript received —-; revised —.

multi-robot battery exchange systems of a specific kind. In our setup, there are two kinds of robots: (a) *task robots* that provide services at requested locations; and (b) *delivery robots* that deliver fully charged batteries to the task robots when they are required. As a consequence, task robots are not required to return to their docking stations for recharging. We study combinatorial problems arising in these systems when the task robots and delivery robots have communication through a central controller which can decide their rendezvous locations.

The combinatorial problems that arise in this case involve multiple aspects of resource scheduling, path planning, and temporal coordination that make them more difficult to solve than well-known combinatorial problems studied in operations research. We present several polynomial-time heuristic algorithms for solving these combinatorial problems. Our algorithms are inspired by techniques used in artificial intelligence and the design of approximation algorithms. We demonstrate the performance of our algorithms in simulation and analyze how they scale with increasing size of the multi-robot system.

Several combinatorial problems akin to ours, like multi-robot path planning, persistent surveillance, and battery delivery, have been recently studied in relation to the multiple-Traveling Salesman Problem (m-TSP) and the Vehicle Routing Problem (VRP) [18], [19]. However, these studies mostly involve formulating these NP-hard problems as Mixed Integer Programs (MIPs). Kamra *et al.* [19] show that the MIP-based solution methods do not scale well with exact solvers.

Recent works have also proposed heuristics for solving such m-TSP and VRP-based formulations faster [20]–[22]. Sariel *et al.* employ a dynamic task selection scheme to generate priority-based rough schedules for the m-TSP [20]. Yu *et al.* draw connections between multi-agent path planning and network flows, thereby formulating the problem as an Integer Linear Program (ILP) and solving it with principled heuristics [22]. Although these works provide efficient heuristics to solve the m-TSP and/or the VRP, they do not address the rich temporal aspect of our combinatorial problem that is required for coordination between the various task and delivery robots. They also do not reason about the capacity constraints imposed on delivery robots that set an upper bound on how many batteries each of them can physically carry.

Mathew *et al.* have considered a battery exchange problem similar to ours, but they discretize their state space and have fixed paths for task robots requiring periodic recharge [21], [23]. Our task robots have continuous non-trivial paths which change with service locations, hence our recharging times do not admit any periodic structure. We additionally have capacity constraints on the delivery robots and use fast planning heuristics instead of relying on ILP formulations. Our temporal planning component guarantees the fastest possible coordination (i.e., the minimum horizon) as opposed to the approximate fixed-horizon/receding-horizon planner in [21].

Similar combinatorial problems have also been studied in the multi-robot coverage literature [24]–[26] but they neither impose capacity constraints nor require temporal coordination. Furthermore, our methods work in continuous space and time, whereas the aforementioned approaches make the problem tractable by discretizing space and/or time.

The rest of the paper is organized as follows. Section II formalizes our problem and discusses some preliminaries required to understand our subroutines used in our battery exchange planning algorithm. Section III describes our subroutines for the makespan minimization objective with multiple robots while accommodating on-board power and capacity constraints. Section IV describes the full battery exchange planning algorithm. Section V provides proofs and analysis for the battery exchange planning algorithm. Section VI describes our simulations and discusses the results obtained. Lastly, we summarize our work and allude to some avenues for future work in Section VII.

II. Problem Formulation

We assume the availability of a fixed number of task robots and a fixed number of delivery robots. Within each group, we assume that the robots are functionally indistinguishable.

Let the task robots be $\{r^t_1, r^t_2, \ldots, r^t_M\}$ and let the delivery robots be $\{r^d_1, r^d_2, \ldots, r^d_N\}$. Let the requested service locations be $\{\ell_1, \ell_2, \ldots, \ell_T\}$. For simplicity of exposition, we assume that there is a single service request at each service location[1]. Service requests can be satisfied only by the task robots; however, task robots have limited on-board power. Let D^t be the distance that a task robot can travel on a fully charged battery before exhausting it completely. We assume that a battery drains with distance traveled (not with time).

The role of the delivery robots is to keep the task robots active by exchanging task robots' drained batteries for fully charged ones whenever needed. This would allow the task robots to make longer service trips without returning to the charging stations. Delivery robots are capable of carrying a small number of task robot batteries on-board. Let this capacity be C. We assume that the delivery robots are smaller and more agile, and consequently, can travel much longer distances than task robots even though the batteries that power them are smaller than those of the task robots. Let D^d be the distance that a delivery robot can travel on a fully charged battery before exhausting it completely. We further assume that a delivery robot can replenish the batteries of task robots but not those of other delivery robots or of itself. We then study the combinatorial problem of routing and scheduling both task and delivery robots in order to minimize the *makespan*, i.e., the total time taken to complete plan execution.

We develop an algorithm for solving the above core combinatorial problem based on polynomial-time approximation techniques used for variants of the TSP. We consider different combinatorial aspects of our problem one at a time. In the subroutines presented leading up to the final algorithm, we often use K to represent the number of relevant robots in order to generalize their applicability to the task robots as well as the delivery robots.

A. Preliminaries

One of the basic building blocks of our algorithm considers the TSP on a single robot with its own location and the service

[1]If this is not the case, the common location can be assigned a "composite" service request, or duplicate service locations can be dealt with explicitly.

Algorithm II.1 A polynomial-time factor-2 approximation algorithm for the TSP in a metric space.

Input: robot location ℓ_r; service locations $\ell_1, \ell_2, \ldots, \ell_T$.
Output: a tour visiting each service location exactly once.
Construct a complete edge-weighted undirected graph G with nodes $\{\ell_1, \ell_2, \ldots, \ell_T\} \cup \{\ell_r\}$.
Assign the weight on each edge to be the geometric distance between its endpoint locations.
Let M be the MST computed on G.
Let be a depth first search (DFS) tour of the nodes in G starting from ℓ_r that circumnavigates each edge of M twice.
Short-circuit by removing duplicate nodes after their first occurrence.
return .

Algorithm II.2 A polynomial-time factor-2 approximation algorithm for the TSP in a metric space with multiple robots and the objective of minimizing the total distance traveled.

Input: robot locations $\ell_{r_1}, \ell_{r_2}, \ldots, \ell_{r_K}$; service locations $\ell_1, \ell_2, \ldots, \ell_T$.
Output: a tour $_i$ for each robot r_i ($1 \quad i \quad K$) such that each service location is in exactly one tour and the sum of the distances is minimized.
Construct a complete edge-weighted undirected graph G with nodes $\{\ell_1, \ell_2, \ldots, \ell_T\} \cup \{\ell_{r_1}, \ell_{r_2}, \ldots, \ell_{r_K}\}$.
Assign the weight on each edge to be the geometric distance between its endpoint locations.
Create a fictitious node ℓ_0 connected to $\ell_{r_1}, \ell_{r_2}, \ldots, \ell_{r_K}$ each with an edge of weight 0.
Let M be the MST rooted at ℓ_0.
Let $M_1, M_2, \ldots, M_K$ be the trees obtained after deleting the root ℓ_0 from M.
Let $_i$ be a tour of the nodes in M_i starting from ℓ_{r_i} that circumnavigates each edge of M_i twice.
Short-circuit each tour $_i$ by removing duplicate nodes after their first occurrence.
return $_1, _2, \ldots, _K$.

locations embedded in a metric space. It is well known that the distances must satisfy the triangle inequality in order to design any reasonable polynomial-time approximation algorithm for the TSP [27]. Such an assumption is therefore also required for our problem since it involves variants of the TSP as subproblems and is consequently at least as hard as the TSP.

In the inner loop of our algorithm, we use the well-known polynomial-time factor-2 approximation algorithm for the TSP [28]. This algorithm is based on first constructing the minimum spanning tree (MST) over all locations and, in a subsequent phase, short-circuiting a circumnavigation of the MST that traverses each edge twice in opposite directions. Algorithm II.1 presents the details of this procedure. It serves as a factor-2 approximation algorithm since the cost of the optimal tour has to be greater than the cost of M but the cost of the produced tour can only be less than twice the cost of M after short-circuiting[2]. Moreover, the algorithm runs in polynomial time since the MST can be computed easily. While the inner loop of our algorithm uses the procedure in Algorithm II.1, better approximation algorithms for Euclidean TSPs can be used in lieu of it, if required.

For reasoning about the temporal dimension of our problem, we use the framework of the Simple Temporal Problem (STP) [29]. STPs are widely used in temporal reasoning for their expressive power and tractability. An STP is a constraint satisfaction problem with $n+1$ temporal variables representing executing times of events, $X = \{X_0, X_1, \ldots, X_n\}$. It is typically represented as a graph (V, E). Here, $|V| = n+1$ and $v_i \in V$ represents the execution time of the event $X_i \in X$. A directed edge $e = (v_i, v_j) \in E$ is annotated with the bounds $[LB(e), UB(e)]$ indicating that event X_j must be scheduled between $LB(e)$ and $UB(e)$ time units after event X_i. A special node v_0 corresponding to X_0 represents the "beginning of the world" and is set to 0 by convention.

An STP can also be viewed as a collection of difference constraints of the form $v_j \quad v_i \quad w_{ij}$. Such a collection of difference constraints can in turn also be represented graphically as a *distance graph* where the constraint $v_j \quad v_i \quad w_{ij}$ is represented as a directed edge (v_i, v_j) annotated with w_{ij}. An STP is known to be consistent if and only if its distance graph has no negative cost cycles in it [29]. For a consistent STP, a schedule with the minimum makespan can be found in polynomial time using shortest path computations on its distance graph. Because the distance graph can have negative weights, shortest paths on it are commonly computed using the Bellman-Ford algorithm [27].

B. Extension to Multiple Robots: Total Distance Minimization

The above polynomial-time factor-2 approximation algorithm can be easily extended to the case of multiple robots as well. Given T service locations and K robots, each with its own initial location it must eventually return to, we are interested in obtaining a tour for each robot such that each service location is visited exactly once by one of the robots and the sum of the distances traveled by the robots is minimized.

In order to adapt the procedure of Algorithm II.1 to multiple robots, we do the following. We first create a fictitious node ℓ_0 connected to the initial locations of all robots, each with an edge of weight 0. We then build an MST rooted at ℓ_0. After this tree is built, the fictitious node ℓ_0 is removed—leaving behind a tree rooted at each of the initial locations of the robots. Once we have such a tree for each robot, they are independently short-circuited to produce a tour for that robot. Algorithm II.2 shows this procedure.

Algorithm II.2 runs in polynomial time, and by the earlier argument, provides a factor-2 approximation for the sum of the distances traveled by the robots [30].

III. Subroutine Preparations

We now describe more subroutines required to develop a full battery exchange planning algorithm. These subroutines

[2]Here and in other places, the disambiguation from M also being used to represent the number of task robots is clear from context.

Algorithm III.1 A polynomial-time heuristic algorithm for the TSP in a metric space with multiple robots and the objective of minimizing the makespan.

Input: robot locations $\ell_{r_1}, \ell_{r_2}, \ldots, \ell_{r_K}$; service locations $\ell_1, \ell_2, \ldots, \ell_T$.
Output: a tour τ_i for each robot r_i ($1 \leq i \leq K$) such that each service location is in exactly one tour and the makespan is minimized.
Construct a complete edge-weighted undirected graph G with nodes $\{\ell_1, \ell_2, \ldots, \ell_T\} \cup \{\ell_{r_1}, \ell_{r_2}, \ldots, \ell_{r_K}\}$.
Assign the weight on each edge to be the geometric distance between its endpoint locations.
For each $i \in \{1, 2, \ldots, K\}$, set M_i to be the singleton tree $\{\ell_{r_i}\}$.
Iteratively build trees $M_1, M_2, \ldots, M_K$ as follows until all nodes in G are covered.
- Consider all edges in G with one endpoint in any of $M_1, M_2, \ldots, M_K$ and the other endpoint in neither of them.
- Compute the cost of adding each such edge e as follows.
– Add e to obtain the new set of trees $M_1', M_2', \ldots, M_K'$.
– Let τ_i' be a tour of the nodes in M_i' starting from ℓ_{r_i} that circumnavigates each edge of M_i' twice.
– Short-circuit τ_i' by removing duplicate nodes after their first occurrence.
– The cost of adding e is the cost of the longest resulting tour in $\tau_1', \tau_2', \ldots, \tau_K'$.
- Add the minimum cost edge $e_{\min}$ for the current iteration.
Let τ_i be a tour of the nodes in M_i starting from ℓ_{r_i} that circumnavigates each edge of M_i twice.
Short-circuit τ_i by removing duplicate nodes after their first occurrence.
return $\tau_1, \tau_2, \ldots, \tau_K$.

extend the above algorithms to makespan minimization and accommodate constraints representing limited on-board power and finite capacities of the delivery robots.

A. Extension to Multiple Robots: Makespan Minimization

The above polynomial-time factor-2 approximation algorithm for total distance minimization can be easily extended to makespan minimization as well. Given T service locations and K robots, each with its own initial location it must eventually return to, we are interested in obtaining a tour for each robot such that each service location is visited exactly once and the makespan is minimized under the assumption of uniform velocities of the robots.

In order to adapt the procedure of Algorithm II.2 to makespan minimization, we modify the manner in which the spanning tree M is constructed. Algorithm III.1 shows this procedure. It roots a tree at each initial location of the robots but iteratively grows them in a slightly different way as compared to the algorithm for total distance minimization. In each iteration, it considers all edges that connect a node in any of the partially built trees to a node outside this set. Each such edge is evaluated by how much it would increase the makespan if added. The minimum such edge is chosen

Algorithm III.2 A polynomial-time heuristic algorithm for minimizing the number of recharges required for a single robot having to visit a given set of locations.

Input: robot location ℓ_r; target locations $\ell_1, \ell_2, \ldots, \ell_T$; distance limit D.
Output: a minimum set of tours $\tau_1, \tau_2, \ldots, \tau_K$ to cover the target locations—each tour starting and ending at ℓ_r with a cost $\leq D$.
For $K = 1, 2, \ldots$ do the following.
- Call Algorithm III.1 on $\{\ell_{r_1} = \ell_r, \ell_{r_2} = \ell_r, \ldots, \ell_{r_K} = \ell_r\}$ and $\{\ell_1, \ell_2, \ldots, \ell_T\}$.
- If the minimum makespan is $\leq D$, break.
return $\tau_1, \tau_2, \ldots, \tau_K$ from the last iteration.

to be added for that iteration. An evaluation of the makespan for partially built trees first constructs a tour for each robot circumnavigating the edges of its tree, then short-circuits these tours, and finally uses the cost of the longest resulting tour. As before, when the partially built trees cover all nodes, the trees for the robots are independently short-circuited to produce a final tour for each robot.

Algorithm III.1 runs in polynomial time; but unfortunately, it does not guarantee a factor-2 approximation for the makespan. Nonetheless, it works very well in practice as a heuristic procedure and is used in the auction-based robot routing literature [31], [32].

B. Including On-Board Power: Single Robot

We can now consider a simple problem that accommodates limited on-board power. Suppose we are given a set of target locations and a single robot that needs to visit these locations. The robot can travel a maximum distance of D before it must return to its initial location for recharge. We are interested in minimizing the number of recharges required.

This problem is very similar to makespan minimization for multiple robots with the initial locations of all robots being identical. The multiple robots are conceptually analogous to multiple trips of the same single robot at different times. Thus, Algorithm III.2 assumes K robots in the inner loop and runs Algorithm III.1. If the resulting minimum makespan is greater than D, that iteration is deemed unsuccessful. Starting from $K = 1$ and iteratively increasing K in each unsuccessful iteration, we can determine the lowest value of K for a successful iteration that would be indicative of the number of recharges required.

It is easy to note that a solution exists if and only if all target locations lie within a radius of $D/2$ around ℓ_r. In such a case, Algorithm III.2 also finds a solution in polynomial time although it is not guaranteed to be optimal.

C. Including On-Board Power: Multiple Robots

Now suppose we are given a set of target locations and K robots that need to visit these target locations. Suppose each robot can travel a maximum distance of D before having to recharge at its initial location. If we are interested in minimizing the maximum number of recharges any robot

Algorithm III.3 A polynomial-time heuristic algorithm for minimizing the maximum number of recharges required for any robot given multiple target locations and multiple robots.

Input: robot locations $\ell_{r_1}, \ell_{r_2}, \ldots, \ell_{r_K}$; target locations $\ell_1, \ell_2, \ldots, \ell_T$; distance limit D.
Output: a set of tours for each robot to cover the target locations; each tour for r_i starts and ends at ℓ_{r_i} with a cost $\leq D$.
For $p = 1, 2, \ldots$ do the following.
- Create p copies of each robot r_i at the same location ℓ_{r_i}.
- Call Algorithm III.1 on the locations of these pK robots and $\{\ell_1, \ell_2, \ldots, \ell_T\}$.
- If the minimum makespan is $\leq D$, break.
return the p tours for each of the K robots.

requires, we can generalize the procedure in Algorithm III.2 using a simple "phase counter" p. Starting from $p = 1$, and incrementing it by 1 in each iteration, we find the lowest value for it that passes the following check. In phase p, we create p fictitious identical copies of each of the K robots rooted at the same initial location. We then use Algorithm III.1 to check whether the makespan is $\leq D$. If it passes the check, we stop with the current value of p; else, we increment it for the next iteration. Because a tour is produced for each of the pK fictitious robots, p tours are produced for each real robot which in turn can be traversed by that robot in any order. Algorithm III.3 shows the working of this procedure.

It is easy to note that a solution exists if and only if the maximum distance between a target location and any of the initial locations of the robots is less than or equal to $\frac{D}{2}$. In such a case, Algorithm III.3 also finds a solution in polynomial time although it is not guaranteed to be optimal.

D. Incorporating Capacity Constraints for Delivery Robots

In order to model the capacity constraint C of the delivery robots, we can use Algorithm III.3 with a slight modification to its inner procedure. The new inner procedure, built on Algorithm III.1, maintains partially built trees for each robot and adds one more edge to some robot tree in each iteration until all nodes are covered. If a delivery robot has a capacity of carrying C batteries, no more than C nodes should be assigned to it in this iterative process. In other words, we can simply freeze a robot tree—i.e., make it inactive—in Algorithm III.1 after it has been assigned C service locations. The modified procedure that heeds to the capacity constraint appears in Algorithm III.4.

Algorithm III.4 assumes that all target locations are covered before all robot trees are deemed inactive. In other words, it assumes that a delivery robot doesn't have to go back to its initial location for replenishing its own stock of fresh batteries. When this assumption is not true, a straightforward generalization of the same algorithm fixes the issue simply by using the same idea of a "phase counter" as in Algorithm III.3.

Algorithm III.4 A polynomial-time heuristic algorithm for multiple delivery robots of finite battery-carrying capacities and the objective of minimizing the makespan.

Input: robot locations $\ell_{r_1}, \ell_{r_2}, \ldots, \ell_{r_K}$; target locations $\ell_1, \ell_2, \ldots, \ell_T$; battery-carrying capacity C.
Output: a tour τ_i for each robot r_i $(1 \leq i \leq K)$ such that each target location is in exactly one tour of at most C nodes; and the makespan is minimized.
Construct a complete edge-weighted undirected graph G with nodes $\{\ell_{r_1}, \ell_{r_2}, \ldots, \ell_{r_K}\} \cup \{\ell_1, \ell_2, \ldots, \ell_T\}$.
Assign the weight on each edge to be the geometric distance between its endpoint locations.
For each $i \in \{1, 2, \ldots, K\}$, set M_i to be the singleton active tree $\{\ell_{r_i}\}$.
Iteratively build trees $M_1, M_2, \ldots, M_K$ as follows until all nodes in G are covered.
- Consider all edges in G with one endpoint in any active tree of $M_1, M_2, \ldots, M_K$ and the other endpoint in neither of them.
- Compute the cost of adding each such edge e as follows.
– Add e to obtain the new set of trees $M_1', M_2', \ldots, M_K'$.
– Let τ_i' be a tour of the nodes in M_i' starting from ℓ_{r_i} that circumnavigates each edge in M_i' twice.
– Short-circuit τ_i' by removing duplicate nodes after their first occurrence.
– The cost of adding e is the cost of the longest resulting tour in $\tau_1', \tau_2', \ldots, \tau_K'$.
- Add the minimum cost edge e_{min} for the current iteration.
- Let M be the tree that gets a new target location added to it by the addition of e_{min}.
- If M now has C nodes, make it inactive.
Let τ_i be a tour of the nodes in M_i starting from ℓ_{r_i} that circumnavigates each edge in M_i twice.
Short-circuit τ_i by removing duplicate nodes after their first occurrence.
return $\tau_1, \tau_2, \ldots, \tau_K$.

IV. A Full Multi-Robot Battery Exchange Planning Algorithm

We now put together all the pieces for solving a full-fledged version of the multi-robot battery exchange problem. The task robots are denoted by $\{r_1^t, r_2^t, \ldots, r_M^t\}$; and the delivery robots are denoted by $\{r_1^d, r_2^d, \ldots, r_N^d\}$. The service locations are $\{\ell_1, \ell_2, \ldots, \ell_T\}$. D^t is the distance that a task robot can travel on a fresh battery before exhausting it completely. D^d is the distance that a delivery robot can travel on a fresh battery before exhausting it completely. In our model, the battery drains linearly with distance traveled. D^d is generally much larger than D^t since the delivery robots are fast and agile with lower power consumption rates compared to the task robots that are larger, heavier, and consume more power. C_0 is the capacity of a delivery robot. Finally, we assume that the maximum velocity achievable by any robot r is $V^{max}(r)$ and that the robot can control its velocity to be any value in the interval $[0, V^{max}(r)]$.

ing one assigned task after another. It returns to its initial location after all tasks assigned to it have been completed. However, it does not return to its initial location for recharge since the delivery robots supply fresh batteries to it. The delivery robots keep the task robots alive by exchanging their batteries when needed. Note that delivery robots can neither exchange batteries of other delivery robots nor replace their own batteries. Therefore, they must return back to their initial locations for recharge.

A successful algorithm that attempts to minimize the makespan and coordinates battery exchanges between the task robots and delivery robots should empower task robots to accomplish tasks that require travel distances much larger than D^t. We present such an algorithm below. It has a spatial planning component as well as a temporal coordination component.

A. Spatial Planning

The spatial planning component of the algorithm can be described using the following steps.

Step 1: Assign the service locations to the task robots using Algorithm III.1 in an attempt to minimize the makespan. This step produces a tour for each task robot while disregarding its on-board power capabilities. It is deemed as the responsibility of the delivery robots to keep the task robots alive en route their tours.

Step 2: Consider each task robot's tour and the geometric path it traces. On this path, mark off points at intervals of length D^t.

Step 3: The marked locations from the previous step indicate the rendezvous points—locations where the task robots will need fresh batteries—with q_{ij} denoting the j^{th} rendezvous point on the path of the i^{th} task robot.

Step 4: Assign delivery robots to these rendezvous points using Algorithm III.4. More specifically, first assign delivery robots to all first-order rendezvous points (q_{i1} for all $i \in \{1, 2, \ldots, M\}$), then to all second-order rendezvous points (q_{i2} for all $i \in \{1, 2, \ldots, M\}$), and so on, each time calling Algorithm III.4 as a subroutine.

Step 5: Stitch together the resulting paths for each delivery robot into one single long delivery path that involves multiple returns to that robot's initial location.

The above steps produce paths for task robots to follow in order to visit the service locations and also paths for delivery robots to rendezvous with task robots and exchange their batteries whenever needed.

B. Temporal Coordination

Once paths for all of the task robots and delivery robots are computed, their velocities must be coordinated in such a way that the task robots and delivery robots meet at the rendezvous points for battery exchanges. This coordination should also be done in a way that the entire plan finishes as early as possible. Fortunately, we can do this efficiently in the framework of STPs. The temporal coordination component of the algorithm can be described using the following steps.

Step 1: Create an STP instance containing each of the following events: task robots and delivery robots starting from their initial locations, tasks robots visiting service locations, rendezvous of task robots and delivery robots at q_{ij}'s, and task robots and delivery robots returning back to their initial locations.

Step 2: Include a reference event X_0 in the STP representing the "beginning of the world" and set to 0 by convention.

Step 3: Trace the paths of all task robots and delivery robots as obtained from the spatial planning component of the algorithm.

Step 4: Whenever a robot visits two consecutive locations on its path, with corresponding events X_i and X_j in the STP, add a directed edge from X_i to X_j annotated with the bounds $[\frac{L_{ij}}{V^{max}(r)}, 1]$ where L_{ij} is the physical distance between these locations.

Step 5: Obtain the earliest execution time v_i for each X_i in the STP using the Bellman-Ford algorithm on its distance graph.

Step 6: Set the velocity of a robot traveling between two consecutive locations, corresponding X_i and X_j respectively, to $V_{ij}(r) = \frac{L_{ij}}{v_j - v_i}$.

The above steps produce velocity profiles for each of the task robots and delivery robots to follow in order to satisfy the service requests and also meet at rendezvous points for battery exchanges. A formal proof for properties of the temporal coordination algorithm is presented in the following section.

V. Formal Analysis and Proofs

The foregoing sections described several polynomial-time heuristic procedures for generating the paths of the task robots and the delivery robots under various physical constraints. The STP framework was used for their temporal coordination at rendezvous points. This section presents a formal proof that the algorithm produces velocity profiles for each robot to follow that are in conformance with their maximum velocities, as well as a formal proof that the temporal coordination is itself optimal. This section also presents a formal analysis of the algorithm's time complexity, as well as a report on its theoretical properties.

A. Proof of Optimality for Temporal Coordination

Given an STP S with events $X = \{X_0, X_1, \ldots, X_n\}$, let v_i be the node in its distance graph representation G_S that corresponds to the event X_i. We make use of the following two well-established results [29]: (a) An STP S is consistent if and only if G_S has no negative cost cycles; and (b) For a consistent STP S, two particular valid assignments of execution times to all events are given by $\{X_i = d_{0i}\}$ and $\{X_i = -d_{i0}\}$. Here, d_{ij} is the length of the shortest path from v_i to v_j in G_S. These two assignments correspond to the latest and the earliest possible execution times of all events, respectively.

Let S be the STP as defined in steps 1-4 of Section IV-B. By construction, S contains events for each of the following: task robots and delivery robots at their initial locations, task robots visiting service locations, rendezvous of task robots and delivery robots at q_{ij}'s, and the return of all task robots and delivery robots to their initial locations. S also contains edges that encode the constraints $X_j - X_i \leq \frac{L_{ij}}{V^{max}(r)}$ and $X_j - X_i \geq 1$ for every pair of consecutive locations visited by robot

corresponding to the events X_i and X_j, respectively. Consider the following claims:

1) S is consistent;
2) The temporal coordination happens in the earliest possible way, i.e., our assignment $A = \{X_i \mid d_{i0}\}$ is not only consistent but also guarantees that there exists no other consistent assignment A' of execution times to all events in X such that $A'(X_i) < A(X_i)$ for any X_i;
3) The velocity of any robot r while traversing any edge is $V^{max}(r)$.

First, note that the STP framework is used for temporal coordination since it is representationally sufficient. The paths generated by the spatial planning component along with the maximum velocity constraints on the robots traversing them can be encoded as STP constraints in a straightforward manner as shown in steps 1-4 of Section IV-B. In addition, any rendezvous can also be encoded as an STP constraint by enforcing the equality of the time points at which the task robot and the delivery robot reach the corresponding rendezvous point. However, for simplicity of the proof, and without loss of generality, let's assume that an STP constraint is used only to enforce that the task robot arrives no later than the delivery robot at any rendezvous point. This relaxation does not destroy generality since simultaneity of arrivals can be retrieved by simply reducing the velocity of the task robot.

The above claims can now be proved formally as follows. First, from the theory of STPs [29], it is well know that proving Claim 1 would automatically also prove Claim 2. Moreover, since S encodes all the maximum velocity constraints into temporal bounds of the form $[\frac{L_{ij}}{V^{max}(r)}; 1]$ in step 4 of Section IV-B, Claim 3 would also be proved as a consequence of proving Claim 1. In addition, in order to prove Claim 1, it sufces for us to show the existence of some consistent schedule for all the robots.[3] A convenient schedule that establishes consistency of S is as follows: (a) set all task robots to perform their planned tasks; (b) wait until every task robot either completes its assigned tasks or stalls for a fresh battery; (c) schedule delivery robots to deliver fresh batteries to all waiting task robots (at known locations); (d) resume the activities of task robots that have not yet nished their assigned tasks; (e) wait until every task robot either completes its assigned tasks or stalls for the second fresh battery; and so on.

In summary, the above claims were proved using the following two critical arguments: (1) all coordination constraints are STP constraints; and (2) proving the consistency of S, i.e., showing the mere existence of some solution, also proves the optimality of the assignment $A = \{X_i \mid d_{i0}\}$ obtained from its distance graph.

B. Time Complexity Analysis

With one robot and T service locations, Algorithm II.1 works on a graph with $O(T)$ nodes and $O(T^2)$ edges. Hence, Algorithm II.1 can be implemented to run in $O(T^2 \log T)$ time using Prim's algorithm with a binary heap for computing the MST [27]. Algorithm II.2 extends this to K robots and runs in $O((K + T)^2 \log(K + T))$ time.

Algorithm III.1 has $O(T)$ iterations in the outer loop. In the inner loop, it checks $O(T)$ candidate service locations to nd the best one to add for that iteration. For each candidate addition, it checks $O(K + T)$ possible choices. And for each such choice, it takes $O(K + T)$ time for evaluating it through a DFS and short-circuiting procedure. The overall time complexity of Algorithm III.1 is therefore $O(T^2(K + T)^2)$.

In Algorithm III.2, it is generally hard to estimate the number of iterations required for termination since this depends on the value of D and the actual locations of the robot and the targets. However, a worst-case bound can still be provided. In the worst case, all target locations are at a distance $D = d$ from the robot and it might only be able to visit one target location per trip. In such a case, Algorithm III.2 requires at least T calls to Algorithm III.1, hence running in $O(T^5)$ time. Algorithm III.3 extends this to multiple robots and has a worst-case time complexity that can be estimated in a similar manner. It too is $O(T^5)$. Moreover, the worst-case time complexity of Algorithm III.4 matches that of Algorithm III.3 since the capacity of each delivery robot is at least one and we already assume the worst case of a single delivery per trip.

For an STP S with distance graph $G_S = (V; E)$, computing the optimal solution requires the Bellman-Ford algorithm that has a time complexity of $O(|V||E|)$ [27].

We now nally analyze the time complexity of our battery exchange planning algorithm. The full algorithm requires the following: (a) calling Algorithm III.1 for M task robots and T service locations; (b) marking rendezvous points on each task robot's path: let the total number of rendezvous points be R; (c) calling Algorithm III.4 with N delivery robots and R rendezvous points (locations); and (d) solving an STP with $O(M + N + T + R)$ nodes and $O(M + N + T + R)$ edges. Aggregating these pieces, the total time complexity is $O(T^2(M + T)^2 + R^5 + (M + N + T + R)^2)$. A pessimistic upper bound on R assumes that each service location is at the end of the radius $D^d/2$ from the delivery robots; and hence, a task robot delegated for it would need $O(D^d/D^t)$ fresh batteries. This makes R as large as $O(\frac{TD^d}{D^t})$. The total time complexity of the algorithm becomes $O(T^2(M + T)^2 + (\frac{TD^d}{D^t})^5 + (M + N + \frac{TD^d}{D^t})^2)$. Under the realistic assumptions that $T \gg M; N$ and that D^d/D^t is a domain-specific constant $\gg 1$, the time complexity of our battery exchange planning algorithm is simplified to $O(T^5)$.

We note that although the running time of our algorithm is shown to be only polynomial in the initial number of robots and service locations, the foregoing worst-case analysis is rather loose. In fact, actual experiments, as reported in Section VI, conrm this and indicate that our algorithm scales much better in practice compared to a fth-degree polynomial.

C. Effective Coverage Area

One of the primary objectives of our battery exchange planning algorithm is to have the task robots be able to attend to service requests that are located far away from them. If the

[3] We note that such a consistent schedule would be chosen for convenience of the existential proof but wouldn't be the actual output of the algorithm itself since the output schedule would be the optimal one.

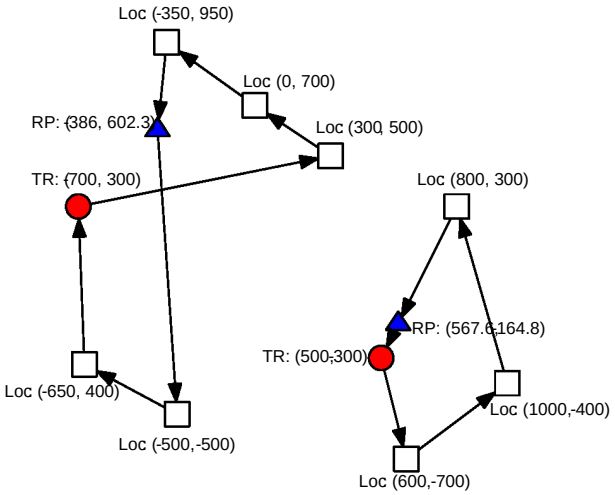

Fig. 1. Shows the spatial plans for task robots. Here, red circles are task robots, blue triangles are locations where task robots need battery exchanges (rendezvous points), and white squares are service locations. A node label indicates its type and its $(x; y)$ coordinates.

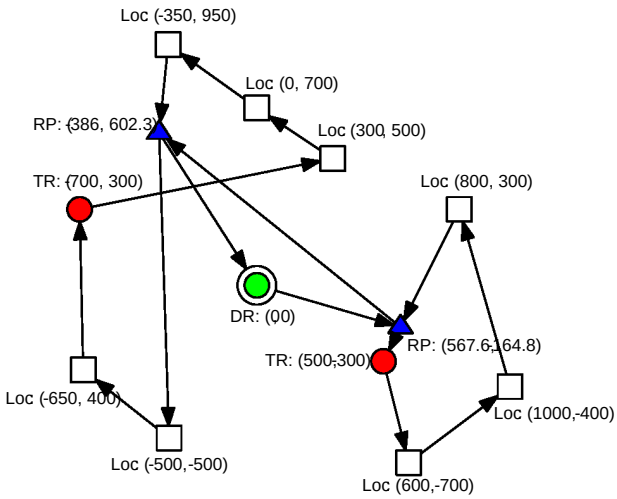

Fig. 2. Shows the spatial plans for delivery robots. Here, red circles are task robots, blue triangles are locations where task robots need battery exchanges (rendezvous points), green double-circles are delivery robots, and white squares are service locations. A node label indicates its type and its $(x; y)$ coordinates.

task robots do not have the support of the delivery robots through the battery exchange planning algorithm, they can only travel a distance of D^t=2 before having to come back to their initial locations for recharge. Through our algorithm, however, the delivery robots can keep the task robots alive at farther distances. The task robots can continue to service requests indefinitely as long as their rendezvous points occur within the union of the circular reachable areas around each of the delivery robots' initial locations. This extends the effective coverage areas by distances up to D^d=2 around the delivery robots' initial locations. Because the delivery robots are faster, more agile, and consume lesser power compared to the task robots, making D^d D^t, our algorithm achieves a significant improvement in the effective coverage area of the task robots.

VI. Simulation Results

In this section, we present simulation results for the battery exchange planning algorithm. Ground robots and quadrotors instantiate the task robots and the delivery robots, respectively. For quadrotors, their path between any two locations is realistically assumed to be a straight line flight path (ignoring takeoff and landing). All experiments were done on a 64-bit machine with Ubuntu 16.04 LTS, 4th Gen. Intel i7 quad-core processor, 2.5 GHz clock speed and 16 GB RAM. Only a single core was used for the experiments (no parallelization was in effect).

The parameters for the delivery robots are similar to those of the AscTec Hummingbird quadrotor: maximum speed $V_d^{\max}$ = 10 m=s and a battery life of approximately 20 min in continuous flight. Thus, delivery robots can travel a maximum distance of about D^d = 12 000 m on a fully charged battery. The battery-carrying capacity of each delivery robot is 2. The task robots are ground robots with parameters similar to the ClearPath Robotics TurtleBot 2: average maximum speed $V_t^{\max}$ = 0:65 m=s and a battery life of 55 min 60 min in continuous motion. Thus, task robots can travel a maximum distance of about D^t = 2160 m on a fully charged battery.

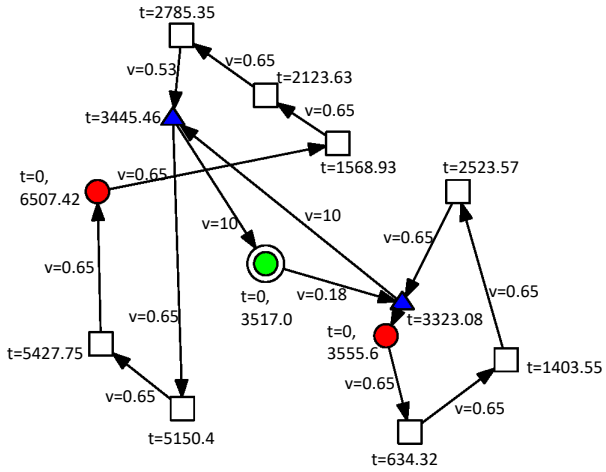

Fig. 3. Shows the complete spatial and temporal plans for all robots. Here, red circles are task robots, blue triangles are locations where task robots need battery exchanges (rendezvous points), green double-circles are delivery robots, and white squares are service locations. A node label indicates the time at which it is visited, and an edge label indicates the velocity of the robot traversing it.

Figures 1–3 provide snapshots of the battery exchange planning algorithm on an example problem instance. In these figures, task robots are indicated using red circles, delivery robots are indicated using green double-circles, rendezvous points are indicated using blue triangles, and service locations are indicated using white squares.

Our example problem instance has 2 task robots, 1 delivery robot, and 8 service locations. Figure 1 shows the result of the spatial planning component of our algorithm. The 2 task robots have been allotted 3 and 5 service locations each by virtue of running Algorithm III.1 while ignoring their maximum traveling distance D^t. It is now deemed as the responsibility of the delivery robot to keep the 2 task robots alive while they carry out their plans. Figure 1 also shows the rendezvous

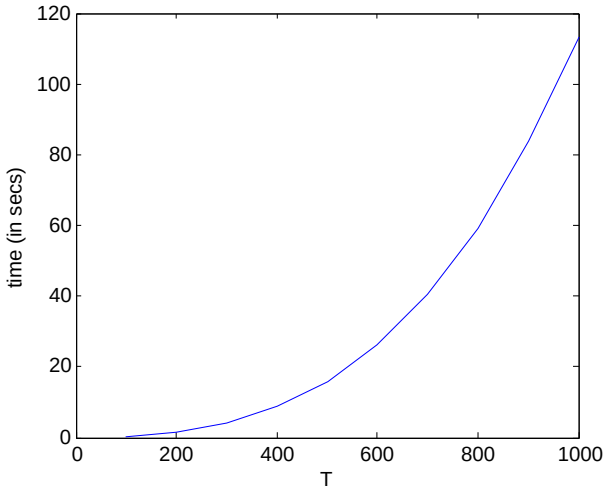

Fig. 4. The effect of T on running time.

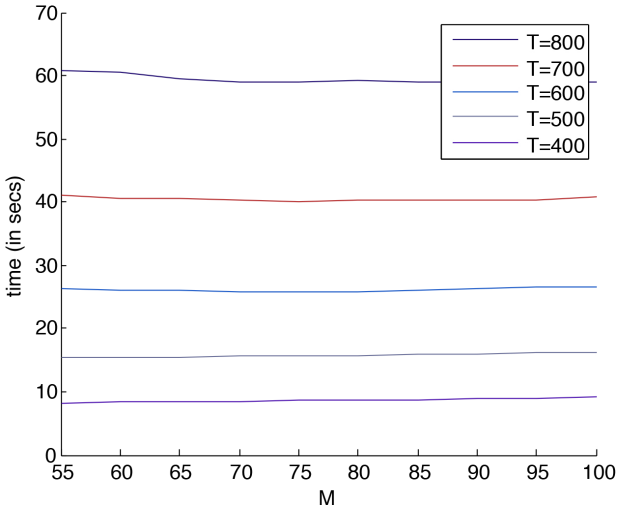

Fig. 5. Shows the effect of M on the running time for different values of T.

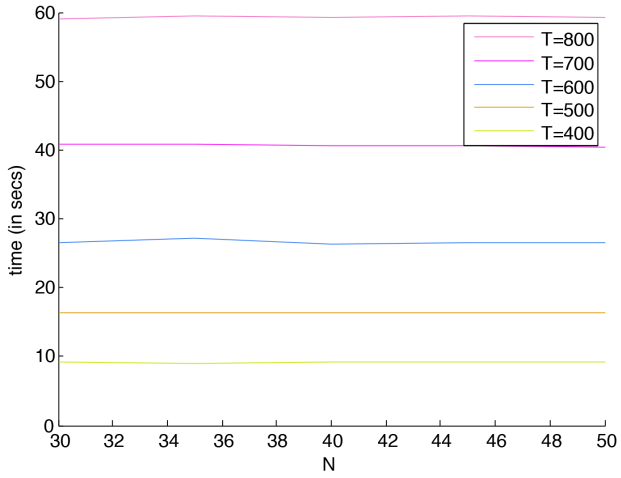

Fig. 6. Shows the effect of N on the running time for different values of T.

points where the task robots will need fresh batteries.

Figure 2 shows the spatial plan for the delivery robot. The rendezvous points act as the destinations for the delivery robot; its visits are scheduled according to Algorithm III.4. The maximum traveling distance of the delivery robot is also taken into account as per the strategy in Algorithm III.3.

Finally, Figure 3 shows the results of the temporal plan computed using the STP framework. The computed visit times are shown next to each node. Here, each edge is also annotated with the velocity of the robot traversing it. It is easy to verify not only that all robots stay within their maximum velocities but also that the temporal coordination happens in the earliest possible way.

Although the spatial planning component of our algorithm is based on approximation techniques and therefore does not guarantee the generation of optimal paths, its temporal planning component is optimal. Moreover, the overall algorithm scales very well to large problem instances in simulation and is therefore likely viable even for real robots.

We now study how the running time of our algorithm scales with increasing the number of task robots, M, the number of delivery robots, N, and the number of service locations, T, while keeping $\frac{D^d}{D^t} = \frac{12000}{2160} = 5{:}55$ fixed. The $(x; y)$ coordinates of the locations and the velocity parameters are generated randomly within suitable ranges.

Figure 4 shows the average running time of the full battery exchange planning algorithm for increasing number of service locations, T. Here, M and N are held constant at 80 and 40, respectively. The running time of the algorithm increases only polynomially with T and not as fast as a fifth-order polynomial (verifiable empirically).

Figures 5 and 6 show the effect of increasing M and N, respectively, on the average running time of the algorithm. In Figure 5, N is held constant at 40; and in Figure 6, M is held constant at 80. The figures indicate that the running times are not significantly affected by M or N, leading us to conclude that the complexity of the algorithm is dominated by T.

Our experimental results confirm our theoretical analysis. Under usual conditions, i.e, when $T \quad M; N$ and $\frac{D^d}{D^t} > 1$, the running time of our algorithm is indeed dominated by T. Furthermore, our experiments indicate a scaling behavior within the scope of—and actually better than—the theoretical worst-case analysis of the time complexity of the algorithm.

VII. Conclusions and Future Work

In this paper, we presented an important approach to solve battery exchange problems for multi-robot systems. Our techniques are broadly applicable to many other combinatorial problems as well, including resource delivery and scheduling, closed-loop product supply management, and pickup and delivery problems for transportation. We addressed several combinatorial aspects of battery exchange problems that are related to path planning, finite on-board energy constraints, capacity constraints on delivery robots carrying batteries, finite maximum velocities, and temporal coordination required between the various task and delivery robots. We presented several heuristic algorithms for solving these problems. Our algorithms are inspired by techniques used in the design of approximation algorithms for other well-know combinatorial

problems such as the m-TSP. Although our algorithms do not guarantee optimality, we conducted experiments to show that they work remarkably well in practice.

Our work essentially proposed an algorithmic approach to solve multi-robot battery exchange problems that is in contrast to the MIP-based modeling approaches popularly used in robotics and operations research. Representationally, our algorithmic approach is more accommodating of complicated domain-specific constraints that are harder to encode in MIP-based models. Furthermore, MIP-based methods generally do not scale well with increasing problem size [19]. However, we showed that the techniques presented in this paper constitute a polynomial-time algorithm that actually scales very well to larger problem instances.

There are many avenues for future work. Although we have shown the effectiveness of our techniques on battery exchange problems with decoupled planning and temporal coordination components, an important avenue for future work is to consider scenarios with a tighter coupling of these components. Such situations might arise when services have associated time window constraints [18]. It would also be interesting to extend our techniques to scenarios with precedence constraints on the services (as in pickup and delivery tasks that have to be done in a certain order). Finally, it would also be interesting to extend our techniques to scenarios where services have rewards associated with them.

REFERENCES

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

[2] J. Derenick, N. Michael, and V. Kumar, "Energy-aware coverage control with docking for robot teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 3667–3672.

[3] B. D. Song, J. Kim, J. Kim, H. Park, J. R. Morrison, and D. H. Shim, "Persistent uav service: An improved scheduling formulation and prototypes of system components," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2013, pp. 915–925.

[4] B. Kannan, V. Marmol, J. Bourne, and M. B. Dias, "The autonomous recharging problem: Formulation and a market-based solution," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 3503–3510.

[5] N. Mathew, S. Smith, and S. Waslander, "A graph-based approach to multi-robot rendezvous for recharging in persistent tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2013, pp. 3497–3502.

[6] S. L. Smith, M. Schwager, and D. Rus, "Persistent monitoring of changing environments using a robot with limited range sensing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5448–5455.

[7] A. Couture-Beil and R. T. Vaughan, "Adaptive mobile charging stations for multi-robot systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 1363–1368.

[8] U. Kartoun, H. Stern, Y. Edan, C. Feied, J. Handler, M. Smith, and M. Gillam, "Vision-based autonomous robot self-docking and recharging," in *IEEE World Automation Congress (WAC)*, 2006, pp. 1–8.

[9] M. C. Silverman, D. Nies, B. Jung, and G. S. Sukhatme, "Staying alive: A docking station for autonomous robot recharging," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2002, pp. 1050–1055.

[10] F. P. Kemper, K. A. Suzuki, and J. R. Morrison, "UAV consumable replenishment: Design concepts for automated service stations," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 369–397, 2011.

[11] T. D. Ngo, H. Raposo, and H. Schioler, "Potentially distributable energy: Towards energy autonomy in large population of mobile robots," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2007, pp. 206–211.

[12] 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, pp. 563–586, 2012.

[13] T. Toksoz, J. Redding, M. Michini, B. Michini, J. P. How, M. Vavrina, and J. Vian, "Automated battery swap and recharge to enable persistent UAV missions," in *AIAA Infotech@ Aerospace Conference*, 2011.

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

[15] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 3335–3340.

[16] T. Toksoz, J. Redding, M. Michini, B. Michini, J. P. How, M. Vavrina, and J. Vian, "Automated battery swap and recharge to enable persistent UAV missions," in *AIAA Infotech@ Aerospace Conference*, 2011.

[17] J. Wu, G. Qiao, J. Ge, H. Sun, and G. Song, "Automatic battery swap system for home robots," *International Journal of Advanced Robotic Systems*, vol. 9, no. 6, 2012.

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

[19] N. Kamra and N. Ayanian, "A mixed integer programming model for timed deliveries in multirobot systems," in *IEEE Conference on Automation Science and Engineering (CASE)*, Aug 2015, pp. 612–617.

[20] S. Sariel-Talay, T. R. Balch, and N. Erdogan, "Multiple traveling robot problem: A solution based on dynamic task selection and robust execution," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, pp. 198–206, 2009.

[21] N. Mathew, S. L. Smith, and S. L. Waslander, "Optimal path planning in cooperative heterogeneous multi-robot delivery systems," in *International Workshop on the Algorithmic Foundations of Robotics*, August 2014.

[22] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016.

[23] N. Mathew, "Discrete path planning strategies for coverage and multi-robot rendezvous," Ph.D. dissertation, University of Waterloo, 2014.

[24] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioğlu, "A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints," *IEEE Transactions on Cybernetics*, vol. 44, no. 3, pp. 305–314, 2014.

[25] M. Kapanoglu, M. Alikalfa, M. Ozkan, and O. Parlaktuna, "A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time," *Journal of Intelligent Manufacturing*, vol. 23, no. 4, pp. 1035–1045, 2012.

[26] G. P. Strimel and M. M. Veloso, "Coverage planning with finite resources," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 2950–2956.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.

[28] V. V. Vazirani, *Approximation Algorithms*. Springer Publishing Company, Incorporated, 2010.

[29] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991.

[30] 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, 2005, pp. 343–350.

[31] S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, and S. Jain, "The power of sequential single-item auctions for agent coordination," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, no. 2, 2006, p. 1625.

[32] S. Koenig, P. Keskinocak, and C. A. Tovey, "Progress on agent coordination with cooperative auctions," in *AAAI*, vol. 10, 2010, pp. 1713–1717.