A Mixed Integer Programming Model for Timed Deliveries in Multirobot Systems

Nitin Kamra and Nora Ayanian

***Abstract*— We present a solution that enables robots to operate in long-duration missions with minimal interruption for recharging or refreshing other resources. Consider a set of deployed "task robots" that request resources (e.g. batteries) from a distribution center, which can deploy "delivery robots" to fulfill those requests. We address the scheduling problem with multiple incoming time-bound requests from the task robots. Our proposed framework incorporates priorities on working robots that can be adjusted by a human operator during scheduling, since mission priorities can change over time. The framework allows a relaxed delivery schedule when available resources are scant and permits dynamic re-routing of delivery robots. The problem is posed as a variant of the Vehicle Routing Problem with Time Windows, and solved as a Mixed Integer Quadratic Program using a branch and bound based solver. We present the specific case of a battery distribution system and validate the results in simulation.**

I. INTRODUCTION

Limited on-board power is a key challenge for mobile robots on long duration deployments. While there has been increasing interest in lifelong deployment of robot teams, literature that addresses the power issue requires robots to return to docking stations to charge or spend significant time charging on-field, both of which can disrupt the mission.

Recent work in persistent robotics include a persistent operation scheduling algorithm that allows Unmanned Aerial Vehicles (UAVs) to return for recharging [1]. Mathew *et al.* allow on-the-spot recharging for persistent task UAVs [2], but assume instantaneous recharging, which is not realistic. Even if charging took longer, this would preclude the charging robot from serving other robots. Song *et al.* replace discharged robots with new ones [3], significantly increasing the number of robots required. Derenick *et al.* propose an energy aware multi-robot system for coverage which alters the team's configuration to allow low-energy robots to return for charging [4]. Kannan *et al.* have introduced the Autonomous Recharging Problem (ARP) and propose an energy-aware design along with static and mobile recharging stations [5]. Other works present persistent monitoring schemes, but ignore battery exhaustion altogether [6].

Recently, hardware-only solutions for recharging and docking mobile robots have been developed [7]–[10]. There has also been recent work on autonomous battery swapping mechanisms specifically for UAVs [11], [12]. We leverage these battery swapping results to address a task scheduling problem for a system which dispatches autonomous robots (the *delivery robots*), from a storage and planning facility (the *control center*) to deliver and recover resources to and from working robots (the *task robots*), that have made a request in order to minimize mission disruption.

The idea of distributable energy was suggested by Ngo *et al.* with a policy for energy sharing [13]. However, they focused on battery sharing among robots to prolong working duration, eventually making them return for recharging.

In this paper we consider scheduling for prioritized timed-delivery requests with multiple capacity-limited delivery robots, and specifically minimize total downtime for task robots. Though we demonstrate a specific application of a battery delivery system, our framework extends easily to all kinds of interchangeable resources.

A. Related Work

We pose the problem as a variant of the Vehicle Routing Problem (VRP). The dynamic VRP has been formulated with a deterministic mixed-integer programming model as well as with stochastic queuing theory based models. We specifically consider a variant called the Vehicle Routing Problem with Time Windows (VRPTW) to model this problem with a mixed-integer programming based model [14].

Mixed-integer programming models have been used for modeling similar problems. Mathew *et al.* address the VRP for a truck with a mounted quadcopter, prove its NP-hardness and solve it as a Generalized Traveling Salesman Problem (TSP) [15]. Karaman *et al.* present a framework for vehicle dispatch that admits complex constraints specified with Metric Temporal Logic [16], but do not take into account carrying capacities of delivery vehicles, enforce an unrealistic fixed return time for recharging delivery vehicles, and do not incorporate timed-visits. Stump *et al.* use the VRPTW model for persistent surveillance with periodic timed visits, but enforce hard constraints on visiting times instead of penalties [17]. Unlike our approach, which allows a relaxed delivery schedule, hard constraints can lead to a breakdown of the system when sufficient surveillance robots are not available. Furthermore, there is no priority for visits and there are no provisions available for human overrides.

Another approach is to use stochastic models of the system and queuing theory. Smith *et al.* model the dynamic VRP with multiple vehicles and two classes of demands [18]. They employ a convex optimization function for delayed deliveries, and propose a routing policy to schedule requests, and analyze bounds on performance of the routing policy. The approach has been extended for multiple classes of stochastic demands, proposing a Separate Queue (SQ) policy

Authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA {nkamra,ayanian}@usc.edu. N. Kamra gratefully acknowledges support from Viterbi Graduate School Ph.D. Fellowship. This work was partially supported by Office of Naval Research grant N00014-14-1-0734.

with merging, analyzing the stability and placing bounds on the service fraction under conditions of heavy load [19].

The Dynamic VRP has also been solved with stochastic demands moving on fixed straight lines for a single delivery vehicle [20], [21]. Bopardikar *et al.* propose the longest path policy to maximize the fraction of timed-requests being served at steady-state and provide analytical lower bounds on service fraction [22]. These stochastic approaches provide a framework to analyze steady state performance of routing policies but in general lack one or more of these desirable features for actual deployment: minimizing down-time of task robots, maximizing delivery throughput in absence of the required number of delivery robots, modeling priorities on task robots, and integrated human control.

B. Contributions

The major contributions of this work are:

- solve the resource delivery and recovery problem with timed requests optimally, using multiple delivery robots with limited on-board power and carrying capacity;
- allow a relaxed schedule (some deliveries can be missed) when all deliveries cannot be made due to lack of resources or delivery robots;
- enable dynamic re-routing of delivery robots enroute;
- impose relative priorities while scheduling when all task robots are not equally important.

II. PROBLEM FORMULATION

Consider a 2-D grid with a control center at the origin $O(0,0)$, m delivery robots and n task robots whose locations are described relative to the control center. By periodically querying the task robots, their locations and battery status can be maintained.

New incoming delivery requests from task robots are collected and added to a list of requests. Requests are processed to keep track of "resource cycles" for robots that periodically need resources with a fixed lifespan e.g., batteries. Such requests can be predicted in advance by tracking the resource life cycle. The request list from task robots containing both actual and predicted (by tracking resource cycles) requests, is always available in the format: [`Robot_ID`, `Location`, `Delivery_time`, `Priority`].

We seek to generate a schedule for delivery robots to serve the maximum number of requests possible, while both minimizing the offset from expected delivery times and taking task robot priorities into account.

A. Assumptions

We assume each delivery robot travels with average speed v. Since v is used only to estimate travel time, delivery robots may move at variable speeds as long as it approximately maintains an average speed v. Though we assume the same average speed for delivery robots, it is straightforward to extend our approach to different average speeds for each delivery robot.

The locations of the moving delivery robots are tracked and their path is extrapolated to estimate their location at the time of the next dispatch (i.e. when computation of the new schedule ends). Task robot locations are considered fixed for the problem, but their locations are tracked and small movements can be handled as pointed out in section IV. We assume there exists at least one path from each delivery robot to every task robot and to the control center.

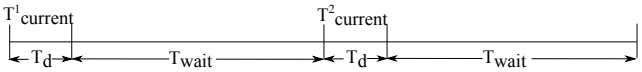


Fig. 1. Timeline of the scheduling algorithm for a single scheduling window; $T^i_{current}$ marks the beginning of computation, T_d is the time for computation so that dispatch begins at $T^i_{current} + T_d$, T_{wait} is the time to wait before computing for the next (shifted) time window.

Lastly we assume sufficient inventory of deliverable resources available at the control center when needed.

B. Graph Representation

We build upon the Vehicle Routing Problem with Time Windows (VRPTW) [14]. The basic idea behind VRPTW framework is to solve a constrained optimization problem which minimizes a penalty function for traveling path costs, subject to the resource distribution constraints.

We introduce an optimization function to combine minimizing traveling costs with penalizing untimely deliveries, in order to incorporate delivery time constraints. The traditional VRPTW imposes delivery timings as hard constraints, which makes the schedule very rigid. By posing the delivery timings as soft constraints in the optimization objective, we have more flexibility of delaying or advancing deliveries to meet stringent delivery times better. This also allows us to miss a few deliveries (if needed) by incurring some penalty, but instead serving other high priority requests on time.

Since future requests are unknown, we cannot schedule for an infinite time horizon. Instead, we solve the optimization problem over a small time window (of length T_W) then continuously repeat the scheduling by shifting the time window by some amount ($\leq T_W$). The time windows are overlapping in nature and later parts of the computed solution for a time window are overwritten every time the window shifts. This enables responding to unforeseen incoming requests and also allows re-routing delivery robots between time windows.

The timeline for scheduling is shown in Fig. 1. Computation for the i^{th} window begins at $T^i_{current}$, the schedule is computed in duration T_d, and robots dispatched at $T^i_{current} + T_d$. The system is then idle for time T_{wait} (used to control scheduling frequency), then repeats for the next window. Time window length (T_W) is chosen as the latest delivery time in the available delivery list (to plan for maximum number of requests), but can be capped at a maximum value if the delivery list contains very late delivery timings. Choosing T_{wait} involves a trade-off between a small value (gives faster response to incoming requests) and a large value (minimizes overlapping window re-computation). We choose T_{wait} as the first delivery time in the delivery list.

Using locations of all $m+n$ robots (extrapolated estimates) and the control center at the origin O, we compute the shortest path between each pair of locations. These could be straight line distances or more complicated paths.

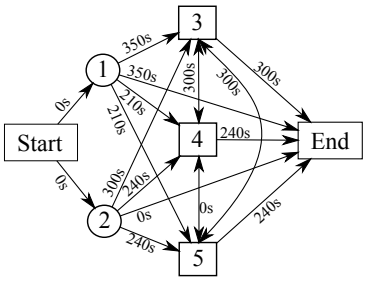


Fig. 2. Example Graph for a time window with two delivery robots (shaped ○) and two task robots (shaped □); node 5 is a copy of node 4 and edge weights represent traveling times (t_{ij}).

For task robots that need multiple deliveries in a single window, we make copies of them at the same location but different delivery timings. In addition, some delivery robots may be recharging and are hence excluded from the current window. For the current window, let $M \leq m$ be the number of delivery robots available and $N \geq n$ be the number of task robots (including copies).

We represent the problem as a graph $G(\mathcal{N}, E)$. Denoting the set of available delivery robots as K ($|K| = M$), the set of task robots (copies included) and available delivery robots together as V ($|V| = M + N$) we add all elements of V to graph G as nodes. We also add to G two nodes located at the control center: `Start` (α) and `End` (ω). Delivery robots are assumed to have begun at the `Start` node. At the end of the window, all delivery robots must return to the `End` node. This rule ensures that if a delivery robot is low on battery or deliverable resources, it will return back to the control center; however, the overlapping windows allow a delivery robot to not return back after each window if charging is not necessary. The full set of $M+N+2$ nodes of G is called $\mathcal{N}$.

Since all paths may not be traversable bidirectionally with the same cost, a directed edge from node i to j is represented as (i, j) with weight t_{ij} representing travel time ($t_{ij} = d_{ij}/v$ where d_{ij} is the shortest distance from node i to j). All task robots have edges leading to each other and to the `End` node. The `Start` node has edges leading to all delivery robots; these edges are considered pre-traversed, so that $t_{\alpha k} = 0, \forall k \in K$. Each delivery robot has an edge to each task robot and to the `End` node. The edge to the `End` node is useful to send a delivery robot directly to the control center or leave it unused if it is already at the control center.

An example with two delivery robots (shaped ○) and two task robots (shaped □) is shown in Fig. 2. Node 5 is a copy of node 4, since it needs two deliveries. Directed edges have weights representing traveling times (t_{ij}).

C. Formulating as an Optimization problem

We now pose a single scheduling window as an optimization problem on the graph G. The decision variables (outputs) of the single window optimization are:

- $x_{ij}^k \in \{0, 1\}$: binary value '1' if k^{th} delivery robot $k \in K$ travels from i^{th} node to j^{th} node and '0' otherwise. The vector of all these variables is denoted x;
- $a_i \in \mathbb{R}^+$: Arrival time at node $i \in (V - K)$. The vector containing all these variables is denoted a;
- $d_i \in \mathbb{R}^+$: Departure time from node $i \in V$. Also the time by which delivery request of node i is considered served. The vector of all these variables is denoted d.

Arrival and departure times are valid only for nodes being visited by some delivery robot, else they are ignored.
We now define the quantities required for optimization:

- t_{ij}: Time to travel from node i to node j.
- τ_i: Time when i^{th} task robot requires delivery.
- T_s: Minimum stay time for a delivery robot to make a delivery; they may stay longer if needed. Can be increased to allow small movements of task robots.
- C^k: Maximum carrying capacity of k^{th} delivery robot.
- c^k: Number of resources already delivered by k^{th} delivery robot during this window.
- B^k: Fraction of total battery power remaining on k^{th} delivery robot.
- B_r^k: Average battery usage per unit distance traveled for the k^{th} delivery robot, as fraction of total battery power.
- $p_i \in [0, 1]$: Priority of task robot i.
- $T_{A,i}$: Advance delivery time for i^{th} task robot.
- Z: A very large positive constant.
- T_{start}: Time at which dispatch begins.
- T_W: Length of current time window.
- t_{bound}: Final time by which all delivery robots must reach the `End` node. We set $t_{bound} = 3T_W/2$.
- λ: Control parameter governing the trade-off in the objective function.

The problem can now be stated as the optimization problem:

$$\min_{x,a,d} \{f_{time}(d) + \lambda f_{travel}(x, a, d)\} \tag{1}$$

where

$$f_{time}(d) = \sum_{i \in (V-K)} p_i (d_i - \tau_i + T_{A,i})^2$$

$$f_{travel}(x, a, d) = \sum_{k \in K} \left(\sum_{\substack{(i,j) \in E \\ j \neq \omega}} x_{ij}^k (a_j - d_i) + \sum_{\substack{(i,j) \in E \\ j = \omega}} x_{ij}^k t_{ij} \right)$$

subject to the constraints:

$$\sum_{k \in K} \sum_{j \in V \cup \{\omega\}} x_{ij}^k \leq 1, \qquad \forall i \in V \tag{2}$$

$$x_{\alpha k}^k = 1, \qquad \forall k \in K \tag{3}$$

$$\sum_{i \in V} x_{i\omega}^k = 1, \qquad \forall k \in K \tag{4}$$

$$\sum_{i \in \{\alpha\} \cup V} x_{ih}^k = \sum_{j \in (V-K) \cup \{\omega\}} x_{hj}^k, \qquad \forall h \in V, k \in K \tag{5}$$

$$a_i - d_i + T_s \leq 0, \qquad \forall i \in (V - K) \tag{6}$$

$$d_i - a_j + t_{ij} \leq Z \left(1 - \sum_{k \in K} x_{ij}^k \right), \quad \forall (i, j) \in E, j \neq \omega \tag{7}$$

$$T_{start} \leq a_i \leq T_{start} + t_{bound}, \qquad \forall i \in (V - K) \tag{8}$$

$$T_{start} \leq d_i \leq T_{start} + t_{bound}, \qquad \forall i \in V \qquad (9)$$

$$T_{start} + t_{bound}\left(1 - \sum_{k \in K} \sum_{j \in V \cup \{\omega\}} x_{ij}^k\right) \leq d_i, \forall i \in V \quad (10)$$

$$d_k - Z(1 - x_{k\omega}^k) \leq T_{start}, \qquad \forall k \in K \qquad (11)$$

$$\sum_{(i,j) \in E} x_{ij}^k - 1 \leq C^k - c^k, \qquad \forall k \in K \qquad (12)$$

$$\sum_{(i,j) \in E} x_{ij}^k B_r^k(t_{ij} v) \leq B^k, \qquad \forall k \in K \qquad (13)$$

$f_{time}()$ is the penalty function for late/early deliveries; d_i is the actual delivery time and $\tau_i - T_{A,i}$ is the expected delivery time for the i^{th} delivery robot. The square function penalizes deviation from the expected delivery time. Any convex penalty function can be chosen, but a square function nicely fits our optimization problem into the Quadratic Programming framework as we will show in Sec. III. The advance time ($T_{A,i}$) ensures delivery slightly before the request time (τ_i). It is useful for resources such as batteries, to prevent full depletion at time τ_i. It also shifts the quadratic penalty function to ensure that early deliveries are penalized less than late deliveries, since though early deliveries are inefficient, they are not as harmful as late deliveries. The priority p_i weighs the penalty imposed by i^{th} node to ensure that higher priority robots impose larger penalties for untimely deliveries. Parameters $T_{A,i}$ and p_i are human controllable for each time window to exert external control.

$f_{travel}()$ assigns a cost to the distance traveled by the delivery robots. For traveling to the `End` node, it uses t_{ij} because a_j is not defined for $j = \omega$. λ controls the trade-off between the two terms of the objective (1).

(2)–(5) define *Path Continuity* constraints for the graph. (2) requires each node i to be visited by either exactly one delivery robot or none. Allowing no visits provides flexibility for a relaxed schedule when all nodes cannot be served in a window due to limited carrying capacity or non-availability of some delivery robots. (3) imposes the pre-traversal from the `Start` node for each delivery robot. (4) requires delivery robots to arrive at the `End` node (ω). (5) requires delivery robot k to exit any node h that it enters.

(6)–(11) define *Time Flow* constraints for the graph. (6) requires departure and arrival times at task robot nodes i to differ by at least T_s, the time needed to deliver/swap a resource. (7) requires for a traversed edge $(i, j) \in E$, that arrival at node j and departure from node i must differ by at least the time required to traverse the edge (t_{ij}). Large positive constant Z ensures that if (i, j) is not being traversed, the constraint is trivially satisfied. (8)–(9) bound arrival and departure times. $t_{bound} > T_W$ gives all delivery robots sufficient time to return back to the control center. (10) coupled with (9), ensures that departure time for a node i that is not visited by any delivery robot is $T_{start} + t_{bound}$. Since all expected delivery times are within time window T_W, this penalizes heavily for not serving task robot i. If the node i is being visited, this constraint is trivially satisfied. (11) ensures that a delivery robot returning directly to the control center starts immediately at the beginning of dispatch. Without this constraint, delivery robots directly returning to the control center will have their departure time set as $T_{start} + t_{bound}$, resulting in stagnation when windows shift.

(12)–(13) impose *Capacity* constraints on the optimization. (12) prevents a delivery robot from delivering more batteries than it is carrying. (13) ensures delivery robots return back to the control center before depleting their own battery. Note that power consumption of delivery robots is based only on traveled distance, while neglecting power required for communication, sensing and processing.

III. SOLVING THE SCHEDULING PROBLEM

We note that the objective function is quadratic and the constraints are purely linear in the decision variables x, a, and d. Since a and d are continuous variables and x is a discrete binary variable, our optimization is a Mixed Integer Quadratic Program (MIQP).

Generally, unconstrained quadratic functions admit a global minimum if the hessian matrix (H) of the objective function is positive definite. But a constrained QP admits a global minimum if the constraints restrict its domain to a non-empty, feasible hyperplane bounded on all sides.

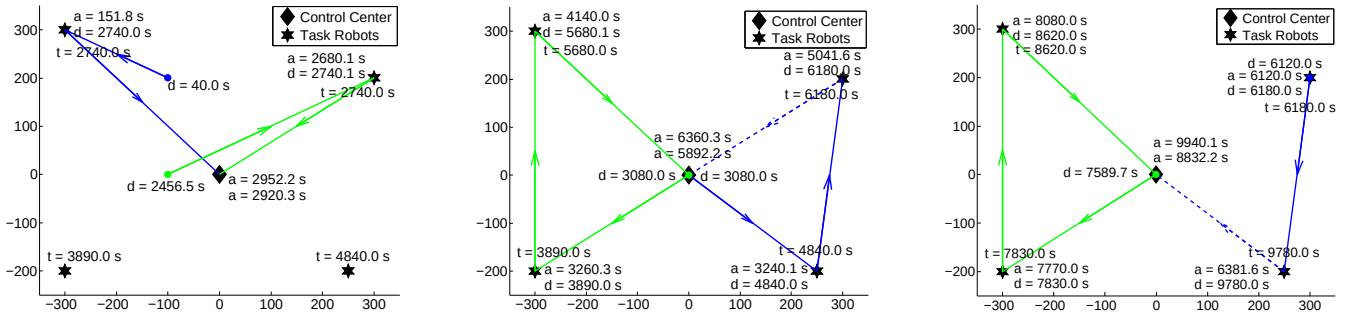
Though in our formulation H is indefinite, our problem domain is fully bounded (since x_{ij}^k is binary, it is bounded; bounds on a_i and d_i are imposed by (8) and (9)). Also note that our constraining equations (eqs. (2)–(13)) can never yield an empty feasible set, hence our MIQP always admits at least one global minimum for any time window.

It is known that a QP is NP-Hard in general (unless H is positive definite) [23] and there exist many methods to solve constrained QPs, e.g., the Augmented Lagrangian, Branch and bound, Conjugate Gradient, and extensions of the Simplex algorithm. We use the SCIP (Solving Constrained Integer Problems) implementation of the OPTI toolbox, based on branch and bound, to solve for global optimum of the MIQP for a single time window [24].

From a single window solution, delivery order and timing is communicated to delivery robots at the next dispatch time ($t = T_{start}$); robots modify their paths accordingly. After time T_{wait} the window shifts, and a new schedule that overrides the previous schedule is computed.

Since this is an NP-hard problem and we use a branch-and-bound approach to obtain the global minimum, the worst-case time required to solve a single window grows exponentially with input size. If there are M available delivery robots and N task robots (including copies) to serve in a window, this formulation results in $M(N^2+N)+3M+2N$ decision variables (joint length of vector a, d, x) and $(M+N)^2 + M(N^2+N) + 11M + 6N + 1$ constraints (eqs. (2)–(13)).

To compute the average computation time for a time window as a function of M, and N, we ran 500 single window simulations, each initialized with random values for M, N, initial locations of task and delivery robots, delivery



(a) Window 1: $[40, 3080]$ seconds (b) Window 2: $[3080, 6120]$ seconds (c) Window 3: $[6120, 9160]$ seconds

Fig. 3. First three schedule windows for two delivery ($\circ$) and four task ($*$) robots; delivery robot paths are marked in their respective colors; arrival and departure time specified at each location (in seconds). (a) First time window: the first scheduled paths with skipped deliveries due to capacity constraints (b) Second time window: next round with all task robots planned to get deliveries (c) Third time window: blue robot gets re-routed for another delivery.

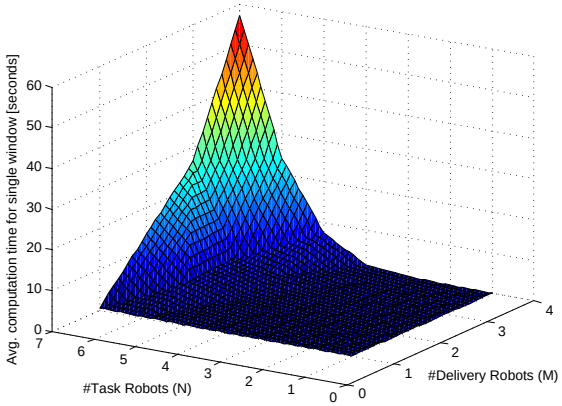


Fig. 4. Average computation time for one window as a function of M, N.

times, remaining capacities, and remaining battery lives, chosen uniformly from reasonable bounded intervals for each parameter. The program was implemented in MATLAB R2012b on a PC with 2.5 GHz Intel Core i7 processor running 64-bit Windows operating system with 16 GB RAM. Computation times were averaged over windows with the same values of (M, N) and are shown in Fig. 4 (interpolated to non-integer values of M and N for clarity of presentation). The results clearly demonstrate the exponential growth in average computation time with increase in M or N.

Due to exponential running time, the current solving approach is viable for small teams of robots, e.g. 2-3 delivery robots and 6-7 task robots. For larger teams, an approximate QP-solver can be used to sacrifice optimality in favor of faster computation time for a single window.

IV. RESULTS

We now present representative simulation results for the case of delivering batteries to ground robots using quadrotors as delivery robots (for whom shortest paths are straight lines). Parameters are based on the AscTec Hummingbird quadrotor: average speed $v = 2\,\mathrm{m/s}$, and battery life approximately $15\,\mathrm{min}$ for continuous flight. Thus they can travel about $1800\,\mathrm{m}$ on full charge, leading to a fractional battery consumption rate of $B_r^k = \frac{1}{1800}$ units/m. Note that delivery robots land and rest without expending energy when not moving, and can hence actually have schedule durations longer than $900\,\mathrm{s}$. Task robots are ground robots with cameras, surveying an area around the control center (at $(0, 0)$), with battery lives of $1-2\,\mathrm{hr}$.

For our simulations, a single window computation can take a maximum of $T_d = 40\,\mathrm{s}$. Replacing a battery takes $30\,\mathrm{s}$ for a delivery robot at the control center and $T_s = 60\,\mathrm{s}$ for a task robot on-site. Note that while we assume task robots' locations to be fixed, this can be relaxed by extending the on-site stay time (T_s) and updating the task robot's location for the following window. Also, once a delivery robot arrives at the control center, it is not scheduled for delivery until the next time window even if its battery has been replaced (we plan to address this in future work).

Figure 3 shows three consecutive schedule windows with two delivery robots (marked $\circ$) and four task robots (marked $*$). Colored lines with arrows show planned paths in each window, with dotted parts being the ones which would be overwritten by next window. All arrival (a), departure (d) and expected delivery times (t) are noted at the respective locations (in seconds). A time window is $T_W = 5000\,\mathrm{s}$ and $T_{wait} = 3000\,\mathrm{s}$, so windows shift by $T_{wait} + T_d = 3040\,\mathrm{s}$, overwriting the remainder of the previous window. Maximum delivery capacity of blue (green) delivery robot is three (two).

In the first window ($[40\,\mathrm{s}, 3080\,\mathrm{s}]$ in Fig. 3(a)) both delivery robots have a single battery left and hence requests from task robots at $(250, -200)$ and $(-300, -200)$ are not fulfilled. This is one of the key strengths of our formulation: if a solution does not exist for the full scheduling problem, it allows missing deliveries by creating a relaxed plan that minimizes the resulting downtime.

In the second window ($[3080\,\mathrm{s}, 6120\,\mathrm{s}]$, Fig. 3(b)) the green delivery robot has full charge and makes two deliveries before returning. The blue delivery robot starts with a full charge and three fresh batteries, though it has been scheduled to deliver only two. It won't make the second delivery before the window shifts. All task robots are planned to receive deliveries for this window.

Figure 3(c) shows the third round of scheduling ($[6120\,\mathrm{s}, 9160\,\mathrm{s}]$). Observe that the blue robot makes its second delivery and, still having a battery left, gets re-routed to make a third delivery before returning. This re-routing

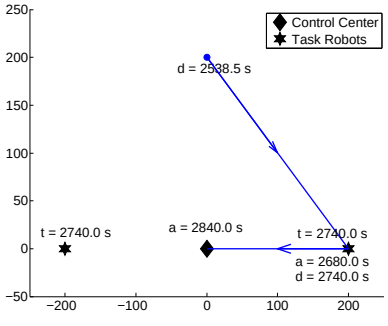


Fig. 5. The effect of priority on deliveries in absence of sufficient resources, carrying capacity or delivery robots.

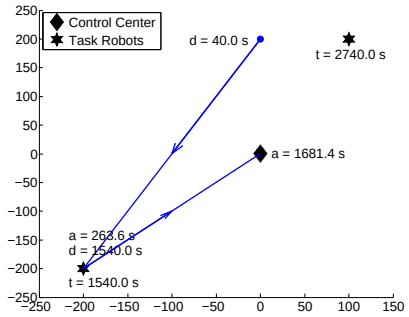


Fig. 6. Counter-intuitive scheduling case to prevent task robot down-time.

across windows is another strength of our formulation and makes better use of time and resources.

Figure 5 shows the effect of priorities, where a delivery robot at $(0, 200)$, with one remaining battery, chooses to deliver to the task robot at $(200, 0)$ over one at $(-200, 0)$ because of its higher priority (everything else being same).

Finally, Fig. 6 has two equal priority task robots, but the delivery robot with a single remaining battery chooses to serve the one farther away due to its earlier delivery time. This occurs since the squared difference $(d_i - (\tau_i - T_{A,i}))^2$ in eqn. (1) will impose a much higher penalty for missing an earlier delivery request (at $\tau_i - T_{A,i}$) than a node with a late delivery time. We justify making the earlier delivery since a later delivery might be served after the window shift, effectively reducing or preventing downtime for both robots.

V. CONCLUSION AND FUTURE WORK

We have proposed a framework enabling task robots to operate in long-duration missions with minimal interruption for recharging, and demonstrated in simulation the specific case of a battery distribution system. The framework deploys robots to serve timed delivery requests and minimizes down-time of robots while taking imposed priorities into account. It allows a relaxed delivery schedule if it is impossible to meet all delivery requests and dynamically re-routes delivery robots by shifting windows.

A limitation of our approach is that, to have an end point of schedule for all delivery robots in a window, they must return to the control center and stay there at least until the time window shifts (regardless of battery swapping time). We plan to address this in future work.

Other future work will include accounting for a fixed inventory of resources and modeling deterioration of resources (due to recycle and reuse) while scheduling.

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] N. Mathew, S. Smith, and S. Waslander, "A graph-based approach to multi-robot rendezvous for recharging in persistent tasks," in *IEEE Intl. Conf. Robot. Autom.*, May 2013, pp. 3497–3502.

[3] B. D. Song, J. Kim, J. Kim, H. Park, and J. Morrison, "Persistent uav service: An improved scheduling formulation and prototypes of system components," in *Intl. Conf. Unmanned Aircraft Systems*, May 2013, pp. 915–925.

[4] J. Derenick, N. Michael, and V. Kumar, "Energy-aware coverage control with docking for robot teams," in *IEEE/RSJ Intl. Conf. Intelligent Robots and Systems*, Sept 2011, pp. 3667–3672.

[5] B. Kannan, V. Marmol, J. Bourne, and M. Dias, "The autonomous recharging problem: Formulation and a market-based solution," in *IEEE Intl. Conf. Robot. Autom.*, May 2013, pp. 3503–3510.

[6] S. Smith, M. Schwager, and D. Rus, "Persistent monitoring of changing environments using a robot with limited range sensing," in *IEEE Intl. Conf. Robot. Autom.*, May 2011, pp. 5448–5455.

[7] A. Couture-Beil and R. Vaughan, "Adaptive mobile charging stations for multi-robot systems," in *IEEE/RSJ Intl. Conf. Intelligent Robots and Systems*, Oct 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 *World Automation Congress*, July 2006, pp. 1–8.

[9] M. Silverman, D. Nies, B. Jung, and G. Sukhatme, "Staying alive: a docking station for autonomous robot recharging," in *IEEE Intl. Conf. Robot. Autom.*, vol. 1, 2002, pp. 1050–1055 vol.1.

[10] F. Kemper, K. Suzuki, and J. Morrison, "Uav consumable replenishment: Design concepts for automated service stations," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 369–397, 2011.

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

[12] 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*, Mar 2011.

[13] T. D. Ngo, H. Raposo, and H. Schioler, "Potentially distributable energy: Towards energy autonomy in large population of mobile robots," in *Intl. Symp. Computational Intelligence in Robotics and Automation*, June 2007, pp. 206–211.

[14] P. Toth and D. Vigo, Eds., *The Vehicle Routing Problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

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

[16] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-uav mission planning," *Intl. Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1372–1395, 2011.

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

[18] S. L. Smith, M. Pavone, F. Bullo, and E. Frazzoli, "Dynamic vehicle routing with heterogeneous demands," in *Proc. IEEE Conf. on Decision and Control*, 2008, pp. 1206–1211.

[19] ——, "Dynamic vehicle routing with priority classes of stochastic demands," *SIAM J. Control and Optimization*, vol. 48, no. 5, pp. 3224–3245, 2010.

[20] S. Bopardikar, S. Smith, F. Bullo, and J. Hespanha, "Dynamic vehicle routing with moving demands - part I: Low speed demands and high arrival rates," in *American Control Conf.*, June 2009, pp. 1454–1459.

[21] S. Smith, S. Bopardikar, F. Bullo, and J. Hespanha, "Dynamic vehicle routing with moving demands - part II: High speed demands or low arrival rates," in *American Control Conf.*, June 2009, pp. 1466–1471.

[22] S. D. Bopardikar, S. L. Smith, and F. Bullo, "On dynamic vehicle routing with time constraints," *IEEE Trans. Robotics*, vol. 30, no. 6, pp. 1524–1532, 2014.

[23] P. Pardalos and S. Vavasis, "Quadratic programming with one negative eigenvalue is np-hard," *Journal of Global Optimization*, vol. 1, no. 1, pp. 15–22, 1991.

[24] T. Achterberg, "Scip: solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009.