Multi-Robot Routing with Rewards and Disjoint Time Windows

Justin Melvin* Pınar Keskinocak* Sven Koenig† Craig Tovey* Banu Yuksel Ozkaya††

*Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

†Computer Science
University of Southern California
Los Angeles, CA 90089

††Industrial Engineering
Hacettepe University
Ankara, Turkey 06800

{jmelvin, pinar, ctovey, byuksel}@isye.gatech.edu skoenig@usc.edu

***Abstract*— Multiple robots are often faster and more fault-tolerant than single robots for applications such as planetary exploration and search and rescue. We study applications where robots move in two-dimensional terrain and have to visit targets of given priorities during given time windows that do not overlap. We analyze the complexity of these coordination tasks and, where possible, use techniques from operations research to develop coordination methods that are efficient and optimize the team performance. We then develop auction-based coordination methods that build on these results and show experimentally that they run in seconds and achieve good team performance for NP-hard coordination tasks.**

I. Introduction

Multiple robots are often faster and more fault-tolerant than single robots for applications such as planetary exploration and search and rescue. We study multi-robot routing problems with rewards and disjoint time windows (that do not overlap), where robots move in two-dimensional terrain and have to visit targets of given priorities during given time windows. For example, robots might have to visit different locations on the surface of the moon to take pictures or collect rock samples. The priority of a location expresses the amount of scientific interest in that location, while its time window expresses when it needs to get visited depending on the sun's location or the temperature of the surface. We first establish the NP-hardness of these multi-robot routing problems. We then discuss special cases where low order polynomial or pseudo-polynomial time methods from operations research achieve optimal team performance, namely minimum cost flow algorithms for identical robots and targets with singleton time windows and dynamic programming for single robots. We then develop auction-based coordination methods for the general case that use dynamic programming for single robots as a subroutine and show experimentally that they run in seconds and achieve good team performance.

II. Assumptions and Notation

The robots move on a connected graph $G = (V, E)$. The set of vertices is $V = R \cup T$, where R denotes the nonempty set of initial vertices of the robots and T denotes the nonempty set of targets. Thus, a vertex refers to either the initial location of a robot or the location of a target. We abuse notation slightly by letting R also represent the set of robots. Each robot $r \in R$ has given time per unit distance $v_r > 0$ and cost per unit distance $c_r > 0$. Each target $t \in T$ has given reward π_t and time window $[a_t, b_t]$, specified by the start time a_t and the end time b_t with $a_t \leq b_t$. The time windows are pairwise disjoint (except possibly at endpoints), by which we mean that no time window intersects the interior of any other time window. The set of edges is $E \subseteq V \times V$. Each edge $(v, v') \in E$ has given distance $d(v, v') = d(v', v) \geq 0$. The distances satisfy the triangle inequality. It can occur that $d(v, v') = 0$ for two vertices $v, v' \in V$ with $v \neq v'$ if, for example, the initial vertex of the robot coincides with a target or two targets share their location.

Each robot $r \in R$ starts at its initial vertex at time zero and then moves from vertex to vertex along the edges of the graph. It takes time $v_r d(v, v')$ and incurs cost $c_r d(v, v')$ for robot $r \in R$ to traverse edge $(v, v') \in E$. The robot can wait an arbitrary amount of time at each target that it visits without incurring any cost. If any robot visits target $t \in T$ in the time window $[a_t, b_t]$, then the robots receive reward π_t for this target once. The objective is to maximize team performance measured in surplus, defined as the sum of the rewards received minus the sum of the costs incurred for traversing edges. We call this problem the multi-robot routing problem with rewards and disjoint time windows. We make use of the following properties:

- If all values v_r, c_r, $d(v, v')$, a_t and b_t are rational, then they can be scaled to integers. We thus assume that they are integers when developing our dynamic programming methods.
- The standard Floyd-Warshall algorithm computes all-pairs shortest paths in $O(|V|^3)$ time and can be used to transform the connected graph G into a complete one. We thus assume that graph G is complete. Then, no loss in optimality is entailed in constraining the robot behavior as follows: If a robot arrives early at a target t, it waits there until the start time a_t. Once a robot is at a target within its time window, it either remains there forever or departs immediately towards a target t' that it can reach before its end time $b_{t'}$ and that is never visited by any other robot.
- A strict total order (= permutation) of the vertices

This research was partly supported by NSF awards under contracts ROBOTICS-0412912, ROBOTICS-0413196, DMI-0113881 and DMI-0457565. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring agency or the U.S. government.

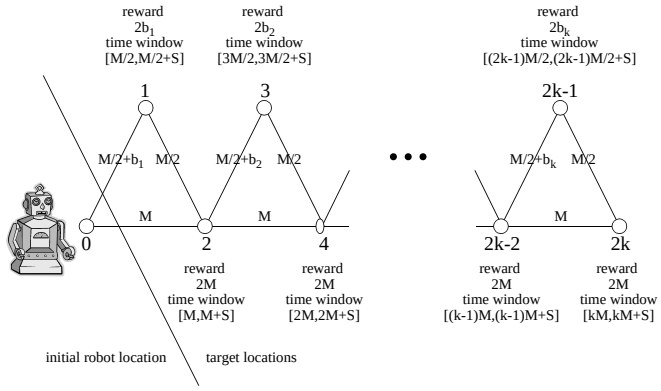

Fig. 1. Example for the NP-Hardness Proof

may be found such that each robot visits vertices in increasing order. Choose any permutation of the targets such that $b_t \leq a_{t'}$ for any two targets t and t' with $t < t'$. Such choice is possible because the time windows are pairwise disjoint. Arbitrarily break ties between singleton time windows (whose start time equals their end time) because the triangle inequality renders their relative ordering irrelevant. Extend the strict total order to all vertices by demanding that $r < t$ for any initial vertex of a robot r and any target t.

III. Related Work

The multi-robot routing problem with rewards and disjoint time windows is related to a number of combinatorial optimization problems from the operations research literature: In the Prize Collecting Traveling Salesman Problem, a reward is associated with each target. The objective is for a single robot (salesperson) to maximize surplus, defined as the sum of the rewards of the targets visited minus the sum of the travel costs [2]. Variants include the Multiple Prize-Collecting Traveling Salesman Problem for multiple robots and the Multiple Prize-Collecting Traveling Salesman Problem with Time Windows, of which our problem is a special case. In the Vehicle Routing Problem with Time Windows, the objective is for multiple robots (vehicles) to minimize the sum of the travel costs but it is required that all targets are visited within their time windows [9].

There is much room for progress in solving these problems. Most solution methods from the literature involve genetic algorithms, tabu search or branch and bound search. An approximation algorithm for the Prize-Collecting Traveling Salesman Problem with Time Windows where all the targets lie on a line or there exists a bound on the ratio of the size of time windows to the distances is given in [13]. An approximation algorithm in the general metric case is given in [3]. Little work has been done so far on solving Multiple Prize-Collecting Traveling Salesman Problems.

IV. Complexity

We first prove that maximizing surplus for multi-robot routing problems with rewards and disjoint time windows is NP-hard.

Theorem 1: Maximizing surplus for multi-robot routing problems with rewards and disjoint time windows is NP-hard even if there is only a single robot and the vertices are in the Euclidean plane.

Proof: We reduce the subset sum problem to our multi-robot routing problem. In an instance of the subset sum problem, we are given a set of positive integers $\{b_1, \ldots, b_k\}$ and an integer $S < \sum_i b_i$. The question is whether there exists a subset $L \subseteq \{1, \ldots, k\}$ such that $\sum_{i \in L} b_i = S$. Construct the following instance of our multi-robot routing problem for a single robot with both time per unit distance and cost per unit distance equal to one that starts at vertex 0, as shown in Figure 1. The targets are the vertices $1, \ldots, 2k$. Target $2i-1$ has reward $2b_i$ and time window $[(2i-1)M/2, (2i-1)M/2+S]$ and target $2i$ has reward $2M$ and time window $[iM, iM+S]$ for $i = 1, \ldots, k$ for some integer $M \gg \sum_i b_i$. The distance between vertices $2i-2$ and $2i$ is M, the distance between vertices $2i-2$ and $2i-1$ is $M/2+b_i$, and the distance between vertices $2i-1$ and $2i$ is $M/2$ for $i = 1, \ldots, k$. These distances could be Euclidean planar distances if the even vertices were collinear, consistent with Figure 1 (the vertical dimension is not drawn to scale). We claim that there exists a subset L such that $\sum_{i \in L} b_i = S$ if and only if there exists a route that yields a surplus of at least $kM + S$ in the constructed instance of the multi-robot routing problem. "Only if" case: If $\sum_{i \in L} b_i = S$, then the surplus is $kM + S$ if the robot visits in numerical order all even vertices and all odd vertices $2i-1$ with $i \in L$. "If" case: We are given a route whose surplus is at least $kM+S$. Since M greatly exceeds the sum of the rewards of the odd vertices, the route must visit all even vertices to reach a surplus of kM. The surplus can be at most $kM+S$ because the reward $2b_i$ of an odd vertex incurs a proportional time delay b_i. The sum of these time delays must not exceed S, otherwise the reward at even vertex $2k$ can not be collected. Hence, the robot visits all even vertices plus a subset of odd vertices whose time delays sum to S. The corresponding subset L thus satisfies $\sum_{i \in L} b_i = S$. ■

It is then also NP-hard to maximize surplus for any given number of robots since the argument from the proof continues to apply if one robot is placed at vertex 0 and all other robots are placed at a new vertex whose distance to vertex 0 is large. Furthermore, the argument from the proof continues to apply for a single robot if we halve the rewards of the odd vertices and assume that the cost per unit distance is zero. It is thus also NP-hard to maximize surplus for a single robot with time per unit distance one but cost per unit distance zero.

V. Special Cases

The NP-hardness of multi-robot routing problems with rewards and disjoint time windows shows that it is likely impossible to maximize surplus in polynomial time. We therefore study efficient techniques from operations research that maximize surplus for special cases of multi-robot routing problems with rewards and disjoint time windows.

A. Identical Robots + Targets with Singleton Time Windows

We show that minimum cost flow algorithms maximize surplus in polynomial time for multi-robot routing problems with rewards and disjoint time windows if the robots are identical and the targets have singleton time windows, that is, $c_r = c$ and $v_r = v$ for all robots r and $a_t = b_t$ for all targets t. We convert the given multi-robot routing problem into a minimum cost network flow problem. The minimum cost network flow problem has one source node x_r with supply one for each robot r, one sink node q with demand $|R|$, and two nodes y_t and z_t for each target t. It has an edge of cost $d(r,t)c$ from node x_r to node y_t for each robot r and target t with $d(r,t)v \leq a_t$, an edge of cost $d(t,t')c$ from node z_t to node $y_{t'}$ for each pair of targets t and t' with $t < t'$ and $a_t + d(t,t')v \leq a_{t'}$, an edge of cost $-\pi_t$ from node y_t to node z_t for each target t, an edge of cost zero from node x_r to node q for each robot r, and an edge of cost zero from node z_t to node q for each target t. All edges have capacity one. Then, by the integrality property of minimum cost networks, there is a minimum cost flow where the flow along each edge is zero or one. This flow corresponds to a behavior that maximizes surplus for the multi-robot routing problem. A unit flow from vertex x_r to vertex y_t indicates that robot r moves from its initial vertex directly to target t. A unit flow from vertex z_t to vertex $y_{t'}$ indicates that the robot that visits target t moves from target t directly to target t'.

B. Single Robots

We show that dynamic programming maximizes surplus in pseudo-polynomial time for single-robot routing problems with rewards and disjoint time windows. Our dynamic program is shown in Figure 2. It uses the following variables: $S(v,x)$ is the maximum surplus that can be obtained by robot r if it starts at its initial vertex at time 0 and is at vertex v at time x and remains there, including any rewards it obtained at or before time x from vertices v with $v < t$ and all travel costs it incurred enroute from its initial vertex to vertex v. We define the start time of vertex r to be $a_r = 0$. (For completeness, we could define $S(t,-1) = -\infty$ for all targets t.) Equation (1) states that the maximum surplus is zero if the robot remains in its initial vertex. Equation (2) states that the maximum surplus of a robot that is at target t at time x and then remains there is calculated as the maximum of the surplus in case the robot has already been at the target at time $x-1$ and in case the robot moved to the target at time x. The latter surplus is calculated by considering all possible vertices v with $v < t$ from which the robot could have moved directly to target t under the restriction that it entered target t before its end time (to ensure that it receives the reward of the target) and that it left vertex v after its start time (to ensure that it receives the reward of the vertex). There are $O(|T| \max_{t \in T} b_t)$ different variables $S(t,x)$, each one of which can be computed in time $O(|T| \max_{t \in T} b_t)$. Thus, the runtime is $O(|T|^2 (\max_{t \in T} b_t)^2)$.

VI. General Case

Often, one needs to maximize surplus for more general versions of multi-robot routing problems with rewards and disjoint time windows than the ones studied so far. We use integer programming for this purpose. This way, we have a gold standard available in the experimental section even though the runtimes for integer programs are very large even for relatively small problems due to the NP-hardness of the problem.

Our integer program is shown in Figure 3. It is similar to some integer programs for traveling salesperson problems and uses the following variables: x_{ijr} is an indicator (0/1) variable for $i < j$ according to the assumed strict total order that is one if and only if robot r moves from vertex i directly to target j. y_{ir} is an indicator (0/1) variable that is one if and only if robot r visits vertex i. w_{ir} is a non-negative variable that denotes how long robot r waits at target i. M is a large number. Constraints (1) and (2) ensure that every robot starts at its initial vertex. Constraint (3) ensures that every robot enters a target if and only if it visits the target. Constraint (4) ensures that every robot leaves a vertex only if it visits the vertex. (However, a robot does not necessarily leave a vertex that it visits since the vertex might be the last vertex that it visits.) Constraint (5) ensures that every target gets entered at most once by any robot. Constraints (6) and (7) ensure that every robot visits a target during its time window if it visits the target. Finally, constraints (8)-(10) restrict the domains of the variables as described above.

VII. Auctions

Since it is computationally intractable to maximize surplus for multi-robot routing problems with rewards and disjoint time windows in general, we now develop auction-based coordination methods that use the dynamic programming methods for single robots from Section V-B as a subroutine and then show experimentally that they run in seconds and still achieve large surpluses. Auctions are decentralized methods that appear to be a promising way of assigning and re-assigning targets to robots. The robots bid on targets and then visit the targets they win. As the robots discover more about the environment during execution, they run additional auctions to re-optimize the allocation of targets among themselves. Such auction-based coordination systems promise to be efficient in terms of communication (robots communicate only essential information and numeric bids) and computation (robots compute their bids in parallel). Auctions have been studied in artificial intelligence starting with the contract net protocol [15] and subsequently in robotics in the context of Robosoccer [12], box pushing [5], security [6] and mapping [14]. A good overview on the current state of the art of auction-based coordination in robotics is given in [4] and analytical results are given in [11]. An auction-based coordination method for the Vehicle Routing Problem with Time Windows is given in [1].

Auction-based coordination is typically based on multi-round auctions where only one target is won by the highest-bidding robot during each round, until all targets have been

$$S(r,x) = 0 \qquad x \geq 0 \qquad (1)$$
$$S(t,x) = \max(S(t,x-1), \max_{v \in V \,|\, x<t, x \leq b_t, a_v \leq x-d(v,t)v_r} (S(v, x-d(v,t)v_r) + \pi_t - d(v,t)c_r)) \qquad t \in T \,|\, x \geq 0 \qquad (2)$$

Fig. 2. Dynamic Program

maximize $\sum_{i \in T, r \in R} \pi_i y_{ir} - \sum_{i \in V, j \in T, r \in R \,|\, i<j} d(i,j) c_r x_{ijr}$ such that

$$y_{rr} = 1 \qquad r \in R \qquad (1)$$
$$y_{ir} = 0 \qquad i \in R, r \in R \,|\, i \neq r \qquad (2)$$
$$y_{ir} = \sum_{j \in V \,|\, j<i} x_{jir} \qquad i \in T, r \in R \qquad (3)$$
$$y_{ir} \geq \sum_{j \in T \,|\, i<j} x_{ijr} \qquad i \in V, r \in R \qquad (4)$$
$$\sum_{i \in V, r \in R \,|\, i<j} x_{ijr} \leq 1 \qquad j \in T \qquad (5)$$
$$a_i \leq M(1-y_{ir}) + \sum_{j \in T \,|\, j \leq i} w_{jr} + \sum_{j \in V, k \in T \,|\, j<k \leq i} d(j,k) v_r x_{jkr} \qquad i \in T, r \in R \qquad (6)$$
$$M(y_{ir}-1) + \sum_{j \in T \,|\, j \leq i} w_{jr} + \sum_{j \in V, k \in T \,|\, j<k \leq i} d(j,k) v_r x_{jkr} \leq b_i \qquad i \in T, r \in R \qquad (7)$$
$$w_{ir} \geq 0 \qquad i \in T, r \in R \qquad (8)$$
$$x_{ijr} \in \{0,1\} \qquad i \in V, j \in T, r \in R \,|\, i < j \qquad (9)$$
$$y_{ir} \in \{0,1\} \qquad i \in V, r \in R \qquad (10)$$

Fig. 3. Integer Program

won by robots [5], [16], [10]. Recently, robotics researchers have also investigated multi-round auctions where a small number of targets are won during each round [8]. In our case, the auctioneer chooses one or more targets during each round that have not yet been won by any robot. Each robot then bids on these targets. Robot r bids on target t the increase in its maximum surplus if it wins the target. In other words, let $T(r)$ be the set of targets that robot r has won already in previous rounds. Then, robot r bids on target t its maximum surplus for the set of targets $T(r) \cup \{t\}$ minus its maximum surplus for the set of targets $T(r)$, where its surplus of a given set of targets is defined as the the sum of the rewards of the set of targets (which must be a subset of the given set of targets) visited by the robot during their respective time windows minus the sum of the costs incurred by the robot for traversing edges. The robot with the highest bid (which can be negative) wins the corresponding target. Ties are broken in favor of robots that have won the least number of targets, and remaining ties are broken lexicographically by robot index. Then, the process continues until all targets have been won by robots. Each robot then visits a subset of the targets it has won in a way that maximizes its surplus.

We propose three different ways for the auctioneer to choose one or more targets during each round that have not yet been won by any robot:

- **ST-SST (Single Target with Smallest Start Time)**: The auctioneer chooses one of the targets with the smallest start time.
- **ST-LR (Single Target with Largest Reward)**: The auctioneer chooses one of the targets with the largest reward, breaking ties towards targets with the smallest start time.
- **ST-All (All Single Targets)**: The auctioneer chooses all targets, similar to sequential single-item auctions [7].

Under the ST-All (All Single Targets) rule, each robot bids its increase in individual surplus on all single targets. We also study the **PT-All (All Pairs of Targets)** rule. Under the this rule, each robot bids its increase in individual surplus on all single targets as well as pairs of targets for which its increase in individual surplus is at least the sum of its increase in individual surplus of the two vertices individually. The robot with the highest bid wins the corresponding target or targets, and the process continues until all targets have been won by robots. Overall, we expect the runtimes of ST-All to be larger than the ones of ST-LR and ST-SST since the robots need to bid on all targets instead of only the target selected by the auctioneer. We expect the runtimes of PT-All to be even larger since the robots need to bid on pairs of targets in addition to single targets.

VIII. Experiments

We now evaluate the efficiency and effectiveness of the auction-based coordination methods ST-SST, ST-LR, ST-All and PT-All, using the following experimental setups. We always use 10 robots with both time per unit distance and cost per unit distance equal to one:

- **Targets**: In RR50 (Random Rewards with 50 targets) and RR100 (Random Rewards with 100 targets), 50 and 100 targets, respectively, are used whose integer rewards are selected uniformly at random from the interval $[1,50]$. In FR100 (Fixed Rewards with 100 targets), 100 targets are used whose rewards are 25 each. (Note that ST-SST and ST-LR are identical for FR100 since all targets have the same reward.)
- **Distances**: In both CLUSTER and VRANDOM, an area of size 100×100 is used, and the initial locations of the robots are selected uniformly at random from the entire area. In CLUSTER, the area is split into nine squares, and the targets are selected uniformly at random from the upper left and lower right squares. In VRANDOM, the targets are selected uniformly at random from the entire area. In both CLUSTER and VRANDOM, the

TABLE I

EXPERIMENTAL RESULTS

Experimental Setup			Scaled Surplus					Runtime				
Targets	Distances	Time Windows	ST-SST	ST-LR	ST-All	PT-All	IP	ST-SST	ST-LR	ST-All	PT-All	IP
RR50	VRANDOM	NEAR	0.954 (6)	0.930 (1)	0.932 (2)	0.894 (0)	1.000 [8]	0.88	0.87	19.71	194.12	558.25
		FAR	0.936 (5)	0.915 (2)	0.937 (2)	0.864 (0)	1.000 [9]	1.00	1.02	22.76	224.30	107.89
		RANDOM	0.947 (3)	0.950 (3)	0.943 (3)	0.892 (0)	1.000 [9]	0.91	0.92	20.63	202.73	19.30
	DRANDOM	NEAR	0.828 (6)	0.808 (2)	0.814 (1)	0.754 (0)	1.000 [2]	6.48	6.32	282.77	1962.02	5638.50
		FAR	0.642 (1)	0.664 (1)	0.681 (7)	0.635 (0)	1.000 [0]	6.60	6.34	283.56	1878.95	7200.00
		RANDOM	0.658 (6)	0.648 (2)	0.643 (1)	0.609 (0)	1.000 [2]	6.79	6.56	295.11	1551.15	6152.50
	CLUSTER	NEAR	0.901 (2)	0.896 (2)	0.920 (4)	0.917 (3)	1.000 [8]	0.94	0.94	20.41	197.37	1317.80
		FAR	0.922 (5)	0.887 (2)	0.908 (2)	0.901 (0)	1.000 [9]	1.42	1.45	33.07	315.67	309.58
		RANDOM	0.770 (6)	0.757 (1)	0.764 (2)	0.726 (0)	1.000 [9]	6.57	6.28	284.11	1864.50	5811.90
RR100	VRANDOM	NEAR	0.930 (7)	0.893 (0)	0.914 (2)	0.879 (0)	1.000 [4]	7.78	7.10	322.27	1727.18	4777.47
		FAR	0.989 (7)	0.957 (1)	0.956 (0)	0.934 (1)	1.000 [8]	7.89	6.68	292.99	1290.31	2239.12
		RANDOM	0.912 (5)	0.895 (1)	0.906 (3)	0.876 (0)	1.000 [8]	6.53	6.81	298.36	1734.70	2217.41
	DRANDOM	NEAR	0.923 (6)	0.897 (2)	0.885 (0)	0.878 (1)	1.000 [7]	6.53	7.74	346.27	505.35	2238.92
		FAR	0.903 (4)	0.879 (1)	0.876 (0)	0.896 (4)	1.000 [7]	6.78	7.77	346.69	1378.01	2096.82
		RANDOM	0.902 (6)	0.869 (2)	0.864 (1)	0.875 (1)	1.000 [7]	6.72	7.83	346.03	1299.24	1924.15
	CLUSTER	NEAR	0.885 (1)	0.866 (2)	0.885 (2)	0.888 (4)	1.000 [7]	8.23	7.54	324.30	1538.41	2756.02
		FAR	0.898 (4)	0.893 (1)	0.896 (1)	0.900 (3)	1.000 [8]	8.61	7.44	316.27	1023.47	1607.43
		RANDOM	0.870 (4)	0.840 (1)	0.864 (3)	0.836 (1)	1.000 [5]	9.47	8.14	352.19	1224.11	3637.62
FR100	VRANDOM	NEAR	0.904 (6)	0.904 (6)	0.879 (1)	0.879 (2)	1.000 [4]	6.72	6.72	296.17	953.13	4981.36
		FAR	0.915 (6)	0.915 (6)	0.887 (3)	0.871 (0)	1.000 [7]	7.44	7.45	334.71	1604.34	3743.90
		RANDOM	0.916 (6)	0.916 (6)	0.896 (2)	0.873 (1)	1.000 [6]	6.93	6.95	304.91	1140.07	3020.03
	DRANDOM	NEAR	0.884 (9)	0.884 (9)	0.817 (0)	0.780 (0)	1.000 [3]	6.61	6.61	288.34	880.37	5734.04
		FAR	0.866 (6)	0.866 (6)	0.836 (0)	0.858 (3)	1.000 [8]	6.93	6.91	293.58	1011.64	1458.66
		RANDOM	0.938 (8)	0.938 (8)	0.904 (1)	0.831 (0)	1.000 [3]	6.84	6.85	299.85	1080.05	6054.70
	CLUSTER	NEAR	0.837 (2)	0.837 (2)	0.817 (0)	0.870 (7)	1.000 [4]	7.85	7.84	331.64	1499.84	5246.20
		FAR	0.842 (4)	0.842 (4)	0.809 (1)	0.843 (4)	1.000 [8]	11.51	11.42	535.71	587.38	2451.39
		RANDOM	0.939 (8)	0.939 (8)	0.911 (1)	0.844 (0)	1.000 [1]	6.71	6.70	290.48	934.15	6524.81

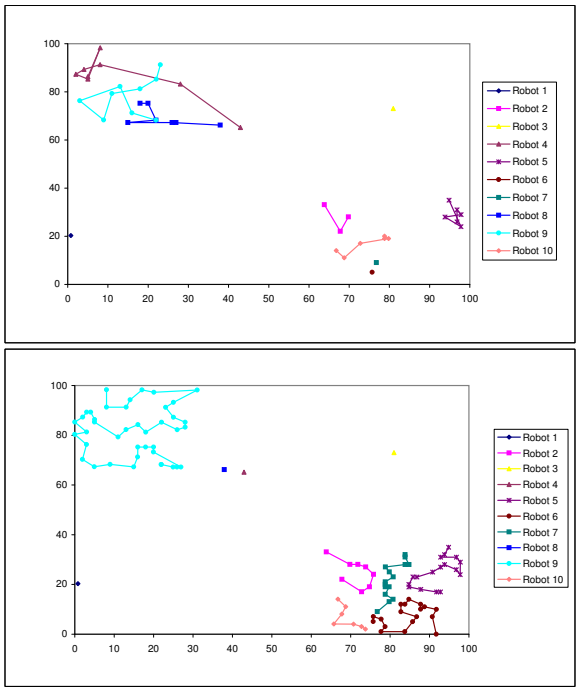

Fig. 4. Example Robot Paths

distances are the straight-line distances between the vertices. In DRANDOM, we create a complete graph with the robots and targets as vertices. We start with all edges having distance 50 and then repeatedly select both an edge and a distance from the interval $[0, 100]$ uniformly at random and make it the distance of the edge if the triangle inequality remains satisfied.

- **Time Windows**: The time windows of the targets are generated by selecting a permutation of the targets. In RANDOM, a permutation of the targets is selected uniformly at random. In NEAR, a permutation of the targets is selected such that targets that are close in distance are close to each other in the permutation. (That is, the next target in the permutation is always the one that is closest to the ten previous targets in the permutation.) In FAR, a permutation of the targets is selected such that targets that are close in distance are far away from each other in the permutation, that is, their time windows are far apart. Thus, the correlation between the closeness of targets in distance and in the permutation is zero, positive and negative for RANDOM, NEAR and FAR, respectively. In all cases, the integer length of the time window of each target is selected uniformly at random from the interval $[0, 10]$. The start time of the first target in the permutation is zero. The start time of all other targets is one larger than the end time of the previous target in the permutation.

The difference in results for RR50 and RR100 helps us to understand the impact of the ratio of targets to robots. The difference in results for RR100 and FR100 helps us to understand the impact of the variability in the rewards. The difference in results for VRANDOM and DRANDOM helps us to understand the impact of the terrain model, namely a "smooth" terrain resulting from an embedding into the Euclidean plane where the distances between pairs of vertices are their straight-line distances versus a "non-smooth" terrain with randomly chosen distances. Finally, the difference in results for combinations of CLUSTER and VRANDOM with RANDOM, NEAR and FAR helps us to understand the

impact of the target proximity in both space and time.

Figure 4 shows an instance for RR100 targets, CLUSTER distances and FAR time windows. The top figure shows the robot paths that maximize surplus. Six robots visit 33 of the 100 targets in this case for a surplus of 1460.08. The other robots do not move. The bottom figure shows the robot paths that maximize surplus if the robots receive the rewards for the targets even if they visit them outside of their time windows (that is, if the time windows were ignored or, equivalently, infinite). Six robots visit all of the 100 targets in this case, but only four of the six robots are the same as in the previous case. However, only 11 targets are visited within their time windows for a surplus (loss) of -248.21 compared to the maximum surplus of 1460.08. This example illustrates that one cannot simply ignore the time windows and hope to achieve a large surplus anyway.

Table I reports the surpluses and runtimes of the auction-based coordination methods ST-SST, ST-LR, ST-All and PT-All as well as the integer program (IP), averaged over nine random instances for each experimental setup, namely combination of targets, distances and time windows. We report all surpluses relative to the surplus of the integer program, which is scaled to one. We report in parentheses for an auction-based coordination method for how many of the nine instances its surplus is larger than the surplus of the other three auction-based coordination methods. We report all runtimes in seconds on a 2.4 GHz Xeon processor PC with 2 GByte of RAM. We solve the integer programs with CPLEX 8.1 with a 2 hour time limit. In case CPLEX cannot determine the maximum surplus within the time limit, we use the upper bound on the maximum surplus reported by CPLEX after 2 hours of runtime instead of the maximum surplus and the time limit instead of the runtime. We report in square brackets for the integer program for how many of the nine instances it finds the maximum surplus. These numbers show that it often does not find the maximum surplus. Thus, we estimate the effectiveness of the auction-based coordination methods very conservatively when we compare their surpluses against upper bounds on the maximum surpluses.

We can say with confidence level 90 percent (using a paired t-test on nine instances) that the surplus of ST-SST is at least as high as that of ST-LR in 20 out of the 27 experimental setups, at least as high as that of ST-All in 15 out of the 27 experimental setups, and at least as high as that of PT-All in 18 out of the 27 experimental setups, which is surprising since its runtime is small. Although the runtime of PT-All is always larger than the one of ST-SST, ST-LR and ST-All, we can say with confidence level 90 percent that its surplus is higher than the surplus of ST-SST, ST-LR and ST-All in only 1, 2 and 3, respectively, out of 27 experimental setups. The runtime of the integer program is the largest by far, yet it does not find the maximum surplus (that is, the optimal solution) within the runtime limit in many cases.

IX. Conclusions

In this paper, we studied multi-robot routing problems with rewards and disjoint time windows. It is important to note that we assumed that the traversal cost and time of each edge are proportional to its distance. This assumption is motivated by our applications but our results immediately generalize to the case where the traversal cost and time of each edge are arbitrary for each robot. It is future work to try out combinatorial bidding rules since they might result in even larger surpluses.

References

[1] J. Antes and U. Derigs. A new parallel tour construction algorithm for the vehicle routing problem with time windows. Technical report, Lehrstuhl fuer Wirtschaftsinformatik und Operations Research, Universitaet zu Koeln, Koeln (Germany), 1995.

[2] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[3] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 166–174, 2004.

[4] M. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7), 2006.

[5] B. Gerkey and M. Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[6] N. Kalra, R. Stentz, and D. Ferguson. Hoplites: A market framework for complex tight coordination in multi-agent teams. Technical Report CMU-RI-TR-04-41, Robotics Institute, Carnegie Mellon University, Pittsburgh (Pennsylvania), 2004.

[7] 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 [nectar paper]. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1625–1629, 2006.

[8] S. Koenig, C. Tovey, X. Zheng, and I. Sungur. Sequential bundle-bid single-sale auction algorithms for decentralized control. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1359–1365, 2007.

[9] A. Kolen, A. Rinnooy Kan, and H. Trienekens. Vehicle routing with time windows. *Operations Research*, 35:266–273, 1987.

[10] M. Lagoudakis, M. Berhault, P. Keskinocak, S. Koenig, and A. Kleywegt. Simple auctions with performance guarantees for multi-robot task allocation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1957–1962, 2004.

[11] M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, S. Koenig, A. Kleywegt, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Proceedings of the International Conference on Robotics: Science and Systems*, pages 343–350, 2005.

[12] R. Nair, T. Ito, M. Tambe, and S. Marsella. Task allocation in the rescue simulation domain: A short note. In A. Birk and S. Coradeschi, editors, *RoboCup 2001: Robot Soccer World Cup V*, Lecture Notes in Computer Science. Springer, 2002.

[13] G. Even R. Bar-Yehuda and S. Shahar. On approximating a geometric prize-collecting traveling salesman problem with time windows. *Journal of Algorithms*, 55:76–92, 2005.

[14] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *Proceedings of the National Conference on Artificial Intelligence*, pages 852–858, 2000.

[15] R. Smith. The contract net protocol: high level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29:1104–1113, 1980.

[16] R. Zlot, A. Stentz, M. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of the International Conference on Robotics and Automation*, pages 3016–3023, 2002.