Multi-Robot Routing with Linear Decreasing Rewards over Time

Ali Ekici and Pınar Keskinocak
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205
{aekici,pinar}@isye.gatech.edu

Sven Koenig
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
skoenig@usc.edu

***Abstract*— We study multi-robot routing problems (MR-LDR) where a team of robots has to visit a set of given targets with linear decreasing rewards over time, such as required for the delivery of goods to rescue sites after disasters. The objective of MR-LDR is to find an assignment of targets to robots and a path for each robot that maximizes the surplus, which is defined to be the total reward collected by the team minus its total travel cost. We develop a mixed integer program that solves MR-LDR optimally with a flow-type formulation and can be solved faster than the standard TSP-type formulations but also show that solving MR-LDR optimally is NP-hard. We then develop an auction-based algorithm and demonstrate that it solves MR-LDR in seconds and with a surplus that is comparable to the surplus found by the mixed integer program with a 12 hour time limit.**

I. Introduction

Teams of robots are more fault-tolerant and faster than single robots. In this paper, we introduce multi-robot routing problems (MR-LDR) where a team of robots has to visit a set of given targets with linear decreasing rewards over time. The objective of MR-LDR is to find an assignment of targets to robots and a path for each robot that maximizes the surplus, which is defined to be the total reward collected by the team minus its total travel cost. Consider, for example, rescue sites in a geographic area that was hit by an earthquake or hurricane. A team of robots has to visit these rescue sites to deliver supplies to the victims. Delays in the deliveries have negative effects, for example, can result in inconvenience, deteriorating health and even death, which is modeled by the linear decreasing rewards over time.

We develop a mixed integer program that solves MR-LDR optimally with a flow-type formulation and can be solved faster than the standard TSP-type formulations. However, we also show that solving MR-LDR optimally is NP-hard. Auction-based algorithms have been used on actual robots [8], [16] and have successfully been applied to simpler kinds of multi-robot routing problems [10], [11], [12], [14]. They are decentralized algorithms and more fault-tolerant than centralized algorithms. They can result in near-optimal solutions and are efficient in terms of both the required amount of computation and communication since information is compressed into numeric bids that the robots can compute in parallel. We therefore develop an auction-based algorithm and demonstrate that it solves MR-LDR in seconds and with a surplus that is comparable to the surplus found by the mixed integer program with a 12 hour time limit.

This research was partly supported by NSF under contracts ITR/AP0113881, IIS-0098807 and IIS-0350584 as well as the US Army Research Laboratory. 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 organizations, agencies, companies or the U.S. government.

II. Problem Definition: MR-LDR

We now define MR-LDR formally. A set of equally fast robots move on a connected undirected graph $G = (V, E)$, where $V = R \cup S$ is the set of vertices, R is the set of initial vertices of the robots and S is the set of vertices of the targets. Each target $s \in S$ has a reward function $f_s(t) = (a_s - tb_s)^+$, where $a_s, b_s > 0$, the ratios $\frac{a_s}{b_s} = d$ are the same for all targets $s \in S$ and x^+ is an abbreviation for $\max(0, x)$. $E \subseteq V \times V$ is the set of edges. Each edge $(s, s') \in E$ has a travel time $d_{ss'} > 0$ and a travel cost $c_{ss'} \geq 0$. We assume that the travel times and travel costs satisfy the triangle inequality.

With a slight abuse of notation, R is also the set of robots. Each robot $r \in R$ starts at its initial vertex at time zero and then moves along the edges of the graph, which takes time and incurs cost. If the first robot visits target $s \in S$ at time t, then the team receives reward $f_s(t) \geq 0$. Note that the reward is zero if and only if $t \geq d$. The objective of MR-LDR is to find an assignment of targets to robots and a path for each robot that maximizes the surplus, which is defined to be the total reward collected by the team minus its total travel cost. The robots might not visit all targets since the rewards become zero at time d and the robots thus have no incentive to move at time d or later.

We make the following assumptions without loss of generality: (1) $S \cap R = \emptyset$ since a target $s \in S$ at an initial vertex of a robot can always be visited at time zero, with reward $f_s(0) = a_s$ and travel cost zero. (2) Graph $G = (V, E)$ is complete since the Floyd-Warshall algorithm can make it complete in time $O(|V|^3)$.

III. Related Work

MR-LDR is related to several combinatorial optimization problems from the literature, as shown in Table I.

In the k-minimum weighted latency problem (k-MWLP), the objective is to find k subtours for k given initial vertices of the robots that minimize the total weighted latency (= visit times of the targets). Thus, k-MWLP is a special case

TABLE I
LITERATURE REVIEW

	Multiple Robots	Rewards	Travel Costs
k-MWLP [15]	yes	always positive	no
MCPTDR [5]	no	becomes zero	no
DR-TSP [3]	no	always positive	no
MR-LDR	yes	becomes zero	yes

of MR-LDR where all reward functions are always positive ($d >> 0$) and all travel costs are zero. The robots visit all targets since all rewards are always positive and all travel costs are zero. k-MWLP is a generalization of the minimum latency problem (MLP) [2], whose objective is to find a tour for a given initial vertex of a single robot that minimizes the total unweighted latency. MLP is at least as hard as the traveling salesperson problem (TSP) [2]. The literature describes numerous approximation algorithms for MLP [1], [2], [9]. The best algorithm is a 3.59-approximation algorithm [4].

In the maximum collection problem with time dependent rewards (MCPTDR), different from MR-LDR, there is only a single robot, the ratios $\frac{a_s}{b_s}$ are not necessarily the same for all targets $s \in S$ and all travel costs are zero. The robot can visit all targets since all travel costs are zero but does not necessarily have to visit all of them. The literature describes an exact branch-and-bound algorithm for MCPTDR and a penalty-based heuristic that finds near-optimal solutions for small MCPTDR instances [5].

In the discounted-reward traveling salesman problem (DR-TSP), different from MR-LDR, there is only a single robot, the fixed rewards are discounted by a constant factor after each time unit and all travel costs are zero. The robot visits all targets since all rewards are always positive and all travel costs are zero. The literature describes a constant-factor $(6.75 + \delta)$ approximation algorithm for DR-TSP [3].

IV. COMPLEXITY

We now explain how difficult it is to solve MR-LDR and some of its special cases optimally. MR-LDR is a combination of TSP (travel costs) and MLP (rewards), two NP-hard problems [13]. For example, solving MR-LDR optimally is NP-hard even if there is only a single robot and all travel costs are zero since MLP is a special case of MR-LDR where there is only a single robot, all reward functions are identical and always positive ($d >> 0$) and all travel costs are zero. We now show that solving MR-LDR optimally remains NP-hard on star graphs even if there is only a single robot and all travel costs are zero.

Theorem 1: Solving MR-LDR optimally is NP-hard on star graphs even if there is only a single robot and all travel costs are zero.

Proof: We reduce from the subset sum problem, which is NP-complete [7]. An instance of the subset sum problem is defined as follows: Given a set of positive integers $A = \{x_1, x_2, \ldots, x_n\}$ and a positive integer X, is there a set $T \subseteq A$ such that $\sum_{x_i \in T} x_i = X$? From such an instance of the subset sum problem, we construct an MR-LDR instance on star graphs where there is only a single robot and all travel costs are zero, see Figure 1. The initial vertex of the robot is 0 (the center of the star graph) and the vertices of the targets are 1, 2, …, n (the terminal vertices of the star graph). Then, $R = \{0\}$ and $S = \{1, 2, \ldots, n\}$, $f_s(t) = (2Xx_s - tx_s)^+$, $d_{0s} = x_s$ and $c_{0s} = 0$. We now show that this MR-LDR instance has a solution with surplus X^2 if and only if the subset sum problem has a solution. Without loss of generality, assume that the robot visits the targets $1, 2, \ldots k$ with $k \leq n$ in order. Then, the robot visits target s at visit time $2\sum_{s'=1}^{s-1} d_{0s'} + d_{0s}$ and the resulting surplus before time $2X$ is

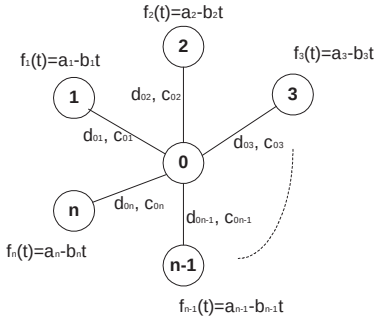

Fig. 1. Star Graph

$$
\begin{aligned}
&\sum_{s=1}^{k}(2Xx_s - (2\sum_{s'=1}^{s-1} d_{0s'} + d_{0s})x_s)^+ \\
&= \sum_{s=1}^{k}(2Xx_s - (2\sum_{s'=1}^{s-1} d_{0s'} + d_{0s})x_s) \\
&= \sum_{s=1}^{k}(2Xx_s - (2\sum_{s'=1}^{s-1} x_{s'} + x_s)x_s) \\
&= 2X\sum_{s=1}^{k} x_s - (2\sum_{s=1}^{k}\sum_{s'=1}^{s-1} x_{s'}x_s + \sum_{s=1}^{k} x_s^2) \\
&= 2X\sum_{s=1}^{k} x_s - (\sum_{s=1}^{k} x_s)^2.
\end{aligned}
$$

Thus, the surplus depends only on $\sum_{s=1}^{k} x_s$. Its maximum value X^2 can be achieved if and only if $\sum_{s=1}^{k} x_s = X$. ■

This special case is important for three reasons: (1) It is a realistic case that models robots that need to return to a depot after visiting a target, for example, to pick up new supplies after they have dropped off their supplies at a rescue site. (2) It shows a difference between k-MWLP and MR-LDR. We have shown that solving MR-LDR optimally on star graphs is NP-hard even if there is only a single robot and all travel costs are zero. On the other hand, k-MWLP can be solved optimally in polynomial time on star graphs if there is only a single robot [15]. (3) It is one of the simplest special cases for which solving MR-LDR optimally is NP-hard. In fact, the following theorem shows that MR-LDR can be solved optimally in polynomial time on star graphs if we impose only one additional restriction, namely that the decrease rates

b_s are the same for all targets $s \in S$.

Theorem 2: MR-LDR can be solved optimally in polynomial time on star graphs if there is only a single robot, $b_s = b$ for every target $s \in S$ and all travel costs are zero.

Proof: Consider an MR-LDR instance on star graphs where there is only a single robot, $b_s = b$ for every target $s \in S$ and all travel costs are zero, see Figure 1. Without loss of generality, assume that $d_{01} \leq d_{02} \leq \ldots \leq d_{0n}$. We propose the following solution: The robot visits the targets $1, 2, \ldots, k^*$ in order, where k^* is the last target that it can visit before time d. Our proposed solution can be found in time $O(n \log n)$. The robot visits its sth target, namely target s, at visit time $t_s^* := 2\sum_{s'=1}^{s-1} d_{0s'} + d_{0s}$. We claim that the surplus cannot be larger if the assignment of targets to the robot or the path of the robot is different. Assume that the robot visits k targets before time d and that it visits its sth target at visit time t_s. $t_s^* \leq t_s$ and $k \leq k^*$ since k^* is the maximum number of targets that the robot can visit before time d. The resulting surplus is

$$\sum_{s=1}^{k} (db - t_s b)^+,$$

whereas the surplus of our proposed solution is

$$\sum_{s=1}^{k^*} (db - t_s^* b)^+ = \sum_{s=1}^{k} (db - t_s^* b)^+ + \sum_{s=k+1}^{k^*} (db - t_s^* b)^+.$$

$(db - t_s^* b)^+ \geq (db - t_s b)^+$ for all $s = 1, 2, \ldots, k$ since $t_s^* \leq t_s$. Furthermore, $(db - t_s^* b)^+ \geq 0$ for all $s = k+1, k+2, \ldots, k^*$, which proves our claim. ■

V. Mixed Integer Program

Although solving MR-LDR optimally is NP-hard, we need to solve at least small MR-LDR instances optimally to have a gold standard. We thus model MR-LDR as a mixed integer program with a flow-type formulation (similar to [6] for MLP):

Maximize
$$\sum_{i \in V} \sum_{j \in S} \sum_{r \in R} (a_j y_{ijr} - d_{ij} x_{ijr} - c_{ij} y_{ijr})$$

subject to

$$\sum_{j \in V} \sum_{r \in R} y_{ijr} \leq 1 \quad i \in V \qquad (a)$$

$$\sum_{i \in V} y_{ijr} = \sum_{k \in V} y_{jkr} \quad j \in V, r \in R \qquad (b)$$

$$\sum_{i \in V} x_{ikr} - \sum_{j \in V} x_{kjr} = b_k \sum_{i \in V} y_{ikr} \quad k \in S, r \in R \qquad (c)$$

$$x_{ijr} \leq M y_{ijr} \quad i \in V, j \in V, r \in R \qquad (d)$$

$$\Big[\sum_{i \in V} \sum_{j \in S} d_{ij} y_{ijr} \leq d \quad r \in R\Big] \qquad (e)$$

$$x_{ijr} \geq 0 \quad i \in V, j \in V, r \in R \qquad (f)$$

$$y_{ijr} \in \{0, 1\} \quad i \in V, j \in V, r \in R. \qquad (g)$$

The variables are: (1) the assignment variable y_{ijr} is one if robot r moves from vertex i directly to vertex j and zero otherwise; and (2) the flow variable x_{ijr} (= the flow of robot r from vertex i directly to vertex j) is the sum of the values b_k of the targets that robot r visits from the time directly before it visits target j until it visits its last target.

The objective is to maximize surplus. Each robot starts at its initial vertex, visits zero or more targets and returns to its initial vertex. The first term in the objective function would be the total reward if the total reward did not decrease over time, namely a reward of a_j for each visited target j. The second term is the decrease of the total reward over time because the total reward of the targets visited by a robot from the time directly before it visits target j until it visits its last target decreases by the flow x_{ijr} times the travel time d_{ij} when it moves from vertex i directly to vertex j. The third term is the total travel cost where we do not count the travel cost when a robot returns to its initial vertex.

The constraints are: (a) a vertex is left by at most one robot; (b) the number of times a vertex is entered by a robot is equal to the number of times it is left by the same robot; (c) the difference of the flow of a robot that enters a target k and leaves the same target is equal to b_k if the robot enters the target and zero otherwise; (d) if a robot does not move from vertex i directly to vertex j then the flow of the robot from vertex i directly to vertex j is zero (M is any constant that is at least as large as the largest flow of any robot, such as $M := \sum_{k \in S} b_k$); (e) the total travel time of a robot is at most d (this inequality is in brackets since it is implied but can be used to strengthen the mixed integer program); (f) flows are non-negative; and (g) assignments are binary.

VI. An Auction-Based Algorithm

Since solving MR-LDR optimally is NP-hard, we need to solve large MR-LDR instances suboptimally. Auction-based algorithms are decentralized algorithms and more fault-tolerant than centralized algorithms. They can result in near-optimal solutions and are efficient in terms of both the required amount of computation and communication since information is compressed into numeric bids that the robots can compute in parallel. We therefore develop an auction-based algorithm for MR-LDR in the following.

A. Ordering Rules

Each robot has a set of targets assigned to it during and after the auction. We now discuss ordering rules that the robot can use to determine in which order to visit these targets to maximize its surplus, where its surplus is the total reward collected by the robot minus its total travel cost. We have already argued that solving MR-LDR optimally is NP-hard even if there is only a single robot. Thus, each robot needs to determine the order greedily in which it should visit the targets. We consider five ordering rules O_1, O_2, …, O_5 and define $w_{ri}(S')$ to be the surplus of robot r and $d_{ri}(S')$ to be its travel time if it orders the targets in S' according to O_i and then visits the targets in this order as long as its surplus increases and does not visit the remaining targets:

O_1: Each robot visits the targets in S' in the order in which

they were assigned to it.[1]

O_2: Each robot pretends that it has not been assigned any targets yet and then assigns itself repeatedly a target with its largest surplus gain per unit of travel time according to O_1 among all targets in S'. Let S_r be the set of targets assigned to robot r, d_r be its travel time and s_r be the last vertex assigned to it. The robot sets $S_r := \emptyset$, $d_r := 0$ and $s_r := r$, and then determines the target $s \in S'$ with its largest surplus gain per time unit. The surplus gain per time unit of target s for robot r is defined as follows (similar to [11]):

$$\frac{w_{r1}(S_r \cup s) - w_{r1}(S_r)}{d_{r1}(S_r \cup s) - d_{r1}(S_r)} = \frac{a_s - b_s(d_r + d_{s_r s}) - c_{s_r s}}{d_{s_r s}}.$$

The surplus gain of a target for a robot is thus the increase in its surplus if it is assigned the target. Then, the robot assigns itself the target s with its largest surplus gain per time unit by setting $S' := S' - s$, $S_r := S_r \cup s$, $d_r := d_r + d_{s_r s}$ and $s_r := s$ and repeats the process until $S' = \emptyset$. It visits the targets in the order in which it assigned them to itself.

O_3: Each robot visits the targets $s \in S'$ in nonincreasing order of their initial rewards a_s.

O_4: Each robot pretends that it has not been assigned any targets yet and then assigns itself repeatedly a target with the largest surplus loss per unit of travel time according to O_1 among all targets in S'. Let S_r be the set of targets assigned to robot r, d_r be its travel time and s_r be the last vertex assigned to it. The robot sets $S_r := \emptyset$, $d_r := 0$ and $s_r := r$, and then determines the target $s \in S'$ with the largest surplus loss per time unit. The surplus loss per time unit of target s for robot r is defined as follows (similar to [5] for MCPTDR), where $w'_{r1}(X, x)$ is an abbreviation for $w_{r1}(X \cup x) - w_{r1}(X)$:

$$\min_{s' \in S' - s} \frac{w'_{r1}(S_r, s) - w'_{r1}(S_r \cup s', s)}{d_{r1}(S_r \cup s) - d_{r1}(S_r)} = \min_{s' \in S' - s} \frac{(-b_s d_{s_r s} - c_{s_r s}) - (-b_s(d_{s_r s'} + d_{s' s}) - c_{s' s})}{d_{r_s s}}.$$

The surplus loss of a target for a robot is thus the decrease in its surplus gain of the target if it visits the best other target in S'. Then, the robot assigns itself the target s with its largest surplus loss per time unit by setting $S' := S' - s$, $S_r := S_r \cup s$, $d_r := d_r + d_{s_r s}$ and $s_r := s$ and repeats the process until $S' = \emptyset$. It visits the targets in the order in which it assigned them to itself.

O_5: Each robot visits the targets in S' according to O_1, O_2, ..., O_4 that results in its largest surplus.

[1]Some of our formulas require one to calculate $w_{r1}(S' \cup s)$ for a target s that has not been assigned to robot r. In this case, we consider target s to be assigned after the targets in S'. Similarly, some of our formulas require one to calculate $w_{r1}(S' \cup s' \cup s)$ for targets s' and s that have not been assigned to robot r. In this case, we consider targets s' and s to be assigned (in this order) after the targets in S'.

B. Auction

All robots use the same two ordering rules, namely O_i during the auction and O_j after the auction (although, technically, O_j is also used during the auction), written as O_i/O_j. O_i has to be run during every round and thus needs to be sufficiently fast, while O_j needs to be run less frequently. The auction then proceeds in several rounds. The robots are the bidders and the targets are the goods. Initially, all targets are unassigned. We consider three bidding rules to determine which unassigned target a robot should bid on during the current round:

B_1: Each robot bids on the unassigned target s with the largest initial reward a_s. Thus, all robots bid on the same target. If at least one bid is submitted, then bidding ends. If no bid is submitted, then the robots bid on the unassigned target with the second-largest initial reward, and so on until the robots bid on the last unassigned target (similar to [12] for multi-robot routing with rewards and disjoint time windows).

B_2: Each robot bids on the one unassigned target s with its largest surplus loss per time unit according to O_1 (Sequential Single-Item Auction Based on Surplus Loss). The surplus loss per time unit of target s for robot r is defined as follows, where S' is the set of unassigned targets, S_r is the set of targets that have already been assigned to robot r in previous rounds and $w'_{r1}(X, x)$ is an abbreviation for $w_{r1}(X \cup x) - w_{r1}(X)$:

$$\min_{s' \in S' - s} \frac{w'_{r1}(S_r, s) - w'_{r1}(S_r \cup s', s)}{d_{r1}(S_r \cup s) - d_{r1}(S_r)}.$$

Thus, each robot might bid on a different target.

B_3: Each robot bids on the one unassigned target with its largest surplus gain per time unit according to O_i (Sequential Single-Item Auction Based on Surplus Gain). The surplus gain per time unit of target s for robot r is defined as follows, where S_r is the set of targets that have already been assigned to robot r in previous rounds:

$$\frac{w_{ri}(S_r \cup s) - w_{ri}(S_r)}{d_{ri}(S_r \cup s) - d_{ri}(S_r)}. \quad (1)$$

Thus, each robot might bid on a different target. (This rule is equivalent to each robot bidding on all unassigned targets.)

Assume that robot r has already been assigned the set of targets S_r in previous rounds and needs to bid on unassigned target s during the current round. It bids its surplus gain per time unit of the target according to Equation (1) if positive and does not bid otherwise.

The largest bid submitted in the current round wins, and the corresponding target is assigned to the corresponding robot. All robots listen to the bids and determine the winning bid in parallel so that no central auctioneer is needed who could become a bottleneck. Another round of the auction

TABLE II

EXPERIMENTAL RESULTS

Experimental Setup				Experiment 1				Experiment 2		Experiment 3		
				Scaled Surplus (Upper Bound of MIP = 1)								
size	d	$c_{ss'}$	targets	B_1 with O_5/O_5	B_2 with O_1/O_5	B_3 with O_5/O_5	MIP	B_1 with O_1/O_5	B_3 with O_1/O_5	B_3 with O_1/O_1	B_3 with O_3/O_3	B_3 with O_4/O_4
5/25	small	small	random	0.865	0.863	0.941	0.999	0.668	0.936	0.926	0.743	0.872
			cluster	0.891	0.849	0.932	0.995	0.756	0.942	0.930	0.748	0.855
		large	random	0.850	0.858	0.940	0.999	0.689	0.934	0.924	0.778	0.875
			cluster	0.908	0.865	0.929	0.985	0.779	0.934	0.922	0.732	0.881
	large	small	random	0.863	0.819	0.848	0.901	0.739	0.847	0.840	0.637	0.845
			cluster	0.881	0.852	0.875	0.923	0.849	0.863	0.856	0.791	0.846
		large	random	0.859	0.811	0.848	0.902	0.734	0.844	0.838	0.636	0.844
			cluster	0.881	0.832	0.875	0.993	0.849	0.864	0.857	0.788	0.834
average				0.875	0.844	0.898	0.955	0.758	0.896	0.887	0.732	0.856
10/50	small	small	random	0.762	0.763	0.794	0.819	0.585	0.792	0.783	0.591	0.769
			cluster	0.699	0.669	0.721	0.736	0.527	0.718	0.705	0.510	0.715
		large	random	0.765	0.765	0.796	0.836	0.586	0.800	0.791	0.614	0.783
			cluster	0.693	0.675	0.725	0.760	0.531	0.718	0.705	0.518	0.704
	large	small	random	0.854	0.845	0.857	0.834	0.788	0.851	0.844	0.731	0.845
			cluster	0.801	0.806	0.829	0.802	0.793	0.812	0.805	0.748	0.817
		large	random	0.851	0.829	0.854	0.864	0.783	0.848	0.841	0.718	0.843
			cluster	0.795	0.807	0.826	0.810	0.785	0.808	0.800	0.735	0.814
average				0.777	0.770	0.800	0.808	0.672	0.793	0.784	0.645	0.786
10/100	small	small	random	0.564	0.594	0.620	0.633	0.362	0.617	0.614	0.341	0.587
			cluster	0.558	0.567	0.602	0.541	0.367	0.600	0.588	0.330	0.550
		large	random	0.557	0.576	0.615	0.629	0.360	0.612	0.609	0.346	0.583
			cluster	0.534	0.554	0.597	0.541	0.363	0.593	0.581	0.314	0.548
	large	small	random	0.702	0.696	0.719	0.594	0.566	0.714	0.709	0.395	0.684
			cluster	0.728	0.734	0.755	0.622	0.662	0.749	0.740	0.607	0.730
		large	random	0.698	0.693	0.720	0.615	0.563	0.711	0.706	0.397	0.688
			cluster	0.718	0.727	0.749	0.594	0.658	0.745	0.737	0.606	0.731
average				0.632	0.643	0.672	0.596	0.488	0.668	0.661	0.417	0.638

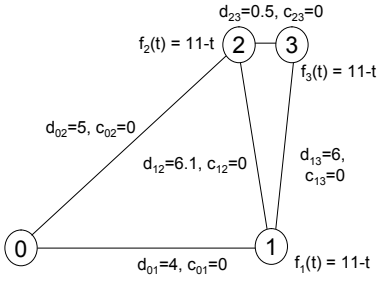

Fig. 2. Example Graph

starts, and the process repeats until robots no longer bid. Then, each robot determines which targets assigned to it not to visit and in which order to visit the remaining targets to maximize its surplus. The robot pretends from now on that the targets assigned to it were assigned to it in the order suggested by O_j. The targets that it does not visit according to O_j become again unassigned. (If each robot performs this step only if it increases its surplus, then the auction is guaranteed to terminate.) Another round of the auction starts, and the process repeats. Eventually, each robot visits the targets assigned to it according to O_j. To understand the purpose of using O_j after the auction, consider the graph from Figure 2 where there is only a single robot and all travel costs are zero. The initial vertex of the robot is 0 and the vertices of the targets are 1, 2 and 3. The robot is assigned the targets in the order 1, 3 and 2 if it uses B_3 with $O_i = O_1$ during the auction. However, its surplus is maximized if it visits the targets in the order 2 and 3. Thus, the robot should not visit all targets assigned to it and should visit the remaining targets in a different order.

VII. EXPERIMENTAL RESULTS

We now evaluate the bidding and ordering rules experimentally in a terrain of size 100×100 for (1) 5 robots and 25 targets (5/25); (2) 10 robots and 50 targets (10/50); or (3) 10 robots and 100 targets (10/100). The initial vertices of the robots are randomly chosen from the entire terrain. The vertices of the targets are randomly chosen from either (1) the entire terrain (random) or (2) the upper left and lower right squares after the terrain was divided into nine equal squares (cluster). We use reward functions with either (1) time limit $d = 50$ (small) or (2) time limit $d = 100$ (large) and decrease rates b_s that are uniformly chosen from $1, 2, \ldots, 20$ for each target s. The travel times are the Euclidean distances between the vertices. The travel costs are either (1) $c_{ss'} = 0.2 d_{ss'}$ (small) or (2) $c_{ss'} = d_{ss'}$ (large). We average over 10 instances for each parameter setting. The large runtime of the mixed integer program prevented us from using a larger sample size.

A. Experiment 1

We first investigate the surplus achieved by the different bidding rules if we use the most sophisticated ordering rule, namely O_5, both during and after the auction. The left columns of Table II evaluate B_1 with O_5/O_5, B_2 with O_1/O_5 (since B_2 has to be used with O_1 during the auction since this ordering rule provides the basis for computing the surplus loss per time unit), B_3 with O_5/O_5 and our mixed-integer program (MIP), that we solve with CPLEX 9.0. The auction-based algorithm solves the MR-LDR instances suboptimally

in seconds. Our mixed-integer program with the flow-type formulation solves them only suboptimally within the run-time limit of twelve hours. (Its optimality gap is about 21.4 percent on average, meaning that its surplus is at most 21.4 percent smaller than optimal on average.) A mixed-integer program with the standard TSP-type formulation (where a binary assignment variable is defined for each combination of target, robot and visit order) cannot solve them at all. (Its optimality gap is about 80 percent on average.) The table reports the scaled surplus as a conservative estimate, where the upper bound of our mixed-integer program has been scaled to one. The surplus of B_3 tends to be larger than the surplus of B_1 and B_2 and can even be larger than the surplus of our mixed-integer program. We learn that B_3 is the best bidding rule with O_5/O_5. Its optimality gap is about 21.0 percent on average.

B. Experiment 2

We now investigate by how much the surplus decreases if we use a less sophisticated ordering rule than O_5 during the auction. The center columns of Table II evaluate B_1 and B_3 with two ordering rules during the auction at opposite ends of the spectrum, namely O_1/O_5 (where the robots assume during the auction that they visit the targets in the order in which they were assigned to them) and O_5/O_5 (where the robots re-order the targets after each round to maximize their surplus approximately). The table does not show B_2 because it cannot be used with O_5 during the auction. Using O_1 during the auction rather than O_5 decreases the surplus by 17.3 percent on average for B_1 and by only 0.6 percent on average for B_3. The surplus decreases the most for B_1 when d is small. We have already identified B_3 as the best bidding rule. We learn that it is sufficient to use B_3 with O_1/O_5 rather than the slower O_5/O_5. Its optimality gap is about 21.4 percent on average.

C. Experiment 3

We now investigate by how much the surplus decreases if we use a less sophisticated ordering rule than O_5 after the auction. The right columns of Table II evaluate B_3 with O_1/O_1, O_2/O_2, ..., O_4/O_4. B_3 with O_1/O_1 and O_2/O_2 results in the same surplus since B_3 assigns the targets in the same order as O_2, namely based on the largest surplus gain per time unit. Using O_1/O_1 rather than O_1/O_5 decreases the surplus by only 1.1 percent on average for B_3. We learn that it is sufficient to use B_3 with O_1/O_1 rather than the slower O_1/O_5. Its optimality gap is about 22.3 percent on average.

VIII. Conclusions

In this paper, we studied multi-robot routing problems (MR-LDR) where a team of robots has to visit a set of given targets with linear decreasing rewards over time, such as required for the delivery of goods to rescue sites after disasters. We developed a mixed integer program that solves MR-LDR with a flow-type formulation and can be solved faster than the usual TSP-type formulations. However, we also showed that solving MR-LDR optimally is NP-hard. We then developed an auction-based algorithm and demonstrated that it solves MR-LDR in seconds and with a surplus that is comparable to the surplus found by the mixed integer program with a 12 hour time limit. We showed that Sequential Single-Item Auctions Based on Surplus Gain (bidding rule B_3) result in a larger surplus than the other bidding rules. It is sufficient if the robots assume during and after the auction that they visit the targets assigned to them in the order in which they were assigned to them (ordering rule O_1) rather than the order that maximizes their surplus approximately but is slower to compute.

References

[1] A. Archer, A. Levin and D. Williamson, "Faster Approximation Algorithms for the Minimum Latency Problem", *in Proceedings of the Symposium on Discrete Algorithms (SODA)*, pp. 88-96, 2003.

[2] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan and M. Sudan, "The Minimum Latency Problem", *in Proceedings of the ACM Symposium on Theory Of Computing (STOC)*, pp. 163-171, 1994.

[3] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson and M. Minkoff, "Approximation Algorithms for Orienteering and Discounted-Reward TSP", *SIAM Journal on Computing*, vol. 37, no. 2, pp. 653-670, 2007.

[4] K. Chaudhuri, B. Godfrey, S. Rao and K. Talwar, "Paths, Trees, and Minimum Latency Tours", *in Proceedins of the Symposium on Foundations of Computer Science (FOCS)*, pp. 36-45, 2003.

[5] E. Erkut and J. Zhang, "The Maximum Collection Problem with Time Dependent Rewards", *Naval Research Logistics*, vol. 43, pp. 749-763, 1996.

[6] M. Fischetti, G. Laporte and S. Martello, "The Delivery Man Problem and Cumulative Matroids", *Operations Research*, vol. 41, no. 6, pp. 1055-1064, 1993.

[7] M. Garey and D. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", 1979, Freeman and Company, New York.

[8] B. Gerkey and M. Mataric, "Sold!: Auction Methods for Multi-Robot Coordination", *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758-768, 2002.

[9] M. Goemans and J. Kleinberg, "An Improved Approximation Ratio for the Minimum Latency Problem", *Mathematical Programming*, vol. 82, pp. 114-124, 1998.

[10] 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 AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1625-1629, 2006.

[11] M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson and S. Jain, "Auction-Based Multi-Robot Routing", *in Proceedings of the International Conference on Robotics: Science and Systems (RoSS)*, pp. 343-350, 2005.

[12] J. Melvin, P. Keskinocak, S. Koenig, C. Tovey and B. Ozkaya, "Multi-Robot Routing with Rewards and Disjoint Time Windows", *in Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2332-2337, 2007.

[13] S. Sahni and T. Gonzales, "P-complete Problems and Approximate Solutions", *in Proceedings of the Annual Symposium on Switching and Automata Theory*, pp. 28-32, 1974.

[14] S. Sariel, T. Balch and J. Stack, "Empirical Evaluation of Auction-Based Coordination of AUVs in a Realistic Simulated Mine Countermeasure Task", *in Proceedings of the International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pp. 197-206, 2006.

[15] B. Wu,"Polynomial Time Algorithms for some Minimum Latency Problems", *Information Processing Letters*, vol. 75, pp. 225-229, 2000.

[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 (ICRA)*, pp. 3016-3023, 2002.