Multi-robot Auctions for Allocation of Tasks with Temporal Constraints

Ernesto Nunes and Maria Gini
Department of Computer Science and Engineering, University of Minnesota
200 Union St SE, Minneapolis, MN 55455
{enunes, gini}@cs.umn.edu

Abstract

We propose an auction algorithm to allocate tasks that have temporal constraints to cooperative robots. Temporal constraints are expressed as time windows, within which a task must be executed. There are no restrictions on the time windows, which are allowed to overlap. Robots model their temporal constraints using a simple temporal network, enabling them to maintain consistent schedules. When bidding on a task, a robot takes into account its own current commitments and an optimization objective, which is to minimize the time of completion of the last task alone or in combination with minimizing the distance traveled. The algorithm works both when all the tasks are known upfront and when tasks arrive dynamically. We show the performance of the algorithm in simulation with different numbers of tasks and robots, and compare it with a baseline greedy algorithm and a state-of-the-art auction algorithm. Our algorithm is computationally frugal and consistently allocates more tasks than the competing algorithms.

Introduction

Auctions are becoming a widely accepted method for allocating tasks to cooperative robots. In an auction allocation method, robots bid on tasks based on the amount of effort needed to complete them. Typically, the effort depends on the distance between the robot location and the task location plus any additional cost for doing the task itself, such as resources consumed (e.g., time spent) in doing the task.

So far, limited attention has been devoted to auctions for allocation of tasks that have to be completed within a specified time window, even if in the real world many tasks have such temporal constraints. For example, a region may need surveillance at regular intervals, a fleet of unmanned aerial vehicles may need to arrive to a task within seconds of each other (Alighanbari, Kuwata, and How 2003), and repairmen have to reach customers within specified time windows.

Time windows make task allocation difficult because the algorithms need to take into account both the spatial and the temporal relationships among the tasks (Kumar, Cirillo, and Koenig 2013). Dealing with overlapping time windows in task allocation remains an open problem (Koenig, Keskinocak, and Tovey 2010).

This paper makes two major contributions:

1. We propose the Temporal Sequential Single-Item auction (TeSSI) algorithm, which allocates tasks using a variant of the sequential single-item auction algorithm. The main cost function used in TeSSI is the *makespan* (i.e., the time the last robot finishes its final task), but we also use (TeSSIduo) a combination of makespan and total distance traveled. Each robot maintains temporal consistency of its allocated tasks using a simple temporal network (STN). TeSSI produces more compact schedules than other algorithms because each robot finds the optimal place in its schedule for each new task before bidding on it. The makespan of the resulting simple temporal network is used to bid. The proposed algorithm supports both offline allocation of tasks, when all the tasks are known upfront, and online allocation, when tasks arrive dynamically.
2. We analyze the algorithm's complexity, and show experimentally that it outperforms a baseline greedy algorithm and the consensus-based bundle auction (CBBA) in experiments with synthetic data and with datasets from vehicle routing problems with time windows (Solomon 1987).

Related Work

Methods for multi-robot task allocation can be broadly categorized into centralized, decentralized, and hybrid; and depending on the optimality of the solution, exact or heuristic. A recent example of a centralized method uses an efficient mixed-integer linear programming approach for multi-robot scheduling with spatial constraints (Gombolay, Wilcox, and Shah 2013). Centralized methods can achieve optimal results, but are not suitable for field operations where communication can be limited and unreliable, and faults are common. Hence, we choose a decentralized approach.

Among decentralized approaches, auctions have enjoyed popularity as a multi-robot coordination method (Dias et al. 2006; Korsah et al. 2010). Auctions move the burden of computation onto individual robots and are robust to local changes or failures, since the auction can proceed with the remaining robots when some robots malfunction (Nanjanath and Gini 2010). In sequential single-item auctions (Lagoudakis et al. 2005; Koenig et al. 2006) one task is allocated at a time, so robots can account for previous commitments while bidding. The auction is cleared in polynomial time producing solutions that are a provable factor away from the optimum for common optimization criteria. Similar guarantees exist for CBBA (Choi, Brunet, and How 2009), which avoids the need for an auctioneer by using a distributed consensus phase.

Auction algorithms have been extended to allocate tasks with temporal constraints. Melvin and colleagues (2007) studied scenarios where tasks have to be completed within a specified time window, but with no overlapping time windows. Assuming that time windows are pairwise disjoint allows robots to choose any order of tasks as long as the robot can reach the next task in time. In our case, such a strict ordering of tasks is not possible, because tasks might have overlapping time windows. Sequential auctions for tasks with overlapping time-windows have been proposed both as single-item (Nunes, Nanjanath, and Gini 2012) and multi-item auctions (Ponda et al. 2010).

Ponda et al. (2010) extended CBBA to tasks with time windows. Time window constraints are accounted for in the robot's bidding function, which is dependent on the robot arrival time. Robots are penalized for arriving late to a task. CBBA has convergence guarantees assuming that the scoring function abides by the principle of diminishing marginal gains. However, this requires the algorithm to not change the start time of the tasks once they are allocated, and thus reduces the number of tasks that the algorithm allocates. Our work overcomes this limitation by allowing tasks' start times to change which results in higher allocation rates.

Problem Definition

We investigate how to allocate tasks with temporal constraints to a group of robots to produce an allocation that satisfies the temporal constraints, and minimizes a cost function. The specific cost function we use is the *makespan* (i.e., the time the last robot finishes its final task), but we also use a combination of makespan and total distance covered.

We assume a nonempty set of robots R and a nonempty set of tasks T. We consider the case when all the tasks are known upfront, and the case when tasks arrive dynamically.

Each $t \in T$ has an earliest start time, ES_t, which is the earliest time t can be executed, a latest start time, LS_t, with $ES_t \leq LS_t$, and a latest finish time LF_t, the time by which t has to be completed. A task also has a duration DUR_t, with $LS_t + DUR_t = LF_t$. $[ES, LF]$ defines the time-window in which the task has to be done.

We use an *auctioneer*, which communicates the tasks to the robots, receives bids from them, decides the allocations, and communicates them to the robots. While the auctioneer can be a single point of failure, it simplifies communication among robots, and can maintain a global view of the individual allocations. Compared to centralized allocation methods, auctions have the advantage of distributing the computation of bids to robots, which allows progression of tasks when a robot becomes disabled or looses communication.

Robots act as *bidders*. Each robot computes the cost of performing each task according to its private schedule. Hence, the underlying optimization function is solved in a decentralized manner by decomposing it into problems that robots solve independently.

TeSSI Algorithm

The auction (Algorithm 1) starts when the auctioneer announces the tasks available for bidding. The auctioneer receives a bid vector from each robot (using the *receiveBid* function). The auctioneer selects the task with the minimum bid among all the bids and allocates that task to the robot that submitted the corresponding bid. The auctioneer notifies all the robots of the winner, ensuring that they all know what tasks have been assigned. Tasks that no robot can do are added to the unallocated tasks set ($T_{unalloc}$). Then the auction restarts with the remaining tasks and continues until the set of tasks is empty.

Algorithm 1: TeSSI algorithm for the auctioneer

```
Input: set of robots $R$, set of unallocated tasks $T$.
1  $T_{unalloc} = \emptyset$ ;
2  while $T \neq \emptyset$ do
3      for $r \in R$ do
4          $sendTasks(r, T)$;
5      for $r \in R$ do
6          $Q_T^r = receiveBid(r)$;
7      $(winner, t_{min}, minBidOverall) = (\text{null,null,}\infty)$;
8      for $r \in R$ do
9          $minBid_r, t = \underset{r,t}{\operatorname{argmin}}\, Q_T^r$;
10         if $minBid_r < minBidOverall$ then
11             $winner = r$; $t_{min} = t$;
12             $minBidOverall = minBid_r$;
13         else if $t_{min} == null$ then
14             $t_{min} = t$;
15     if $winner \neq null$ then
16         $sendWinner(R, winner, t_{min})$;
17     else
18         $T_{unalloc} = T_{unalloc} \cup \{t_{min}\}$;
19     $T = T - \{t_{min}\}$ ;
```

Algorithm 2: TeSSI algorithm for robot r to bid

```
Input: set of unallocated tasks $T$, partial schedule of
       tasks $\bar{T}_r$ for robot $r$
1  $\mathbf{b}_T^r =$
   $\{\ldots, \delta_r^t, \ldots\}$ $\forall t \in T$, where $\delta_r^t = computeBid(t, \bar{T}_r)$;
2  $sendBid(\mathbf{b}_T^r)$;
3  $receiveWinner(R, winner, t_{min})$;
4  if $winner == r$ then
5      $r$ inserts $t_{min}$ in its schedule $\bar{T}_r$;
```

When each robot receives the list of tasks (Algorithm 2), it computes the cost (for instance, makespan) for each task using the *computeBid* function (Algorithm 3) and taking into account its current schedule, which it keeps as a STN. Robot r adds the bid for each task to its vector of bids ($\mathbf{b}_T^r$) and sends it to the auctioneer using *sendBid*. When a robot is notified it won a task, it adds that task to its STN and updates it to keep it consistent.

Managing Schedules

We model the temporal constraints on the tasks as a simple temporal problem (Dechter, Meiri, and Pearl 1991). This model provides a polynomial way for robots to maintain consistent schedules when adding new tasks.

A simple temporal network (STN) is generated as follows: each task is represented by the S_t and F_t time points (illustrated for tasks t_1,t_2 and t_3 in Figure 1). An origin time point is added to the STN, which serves as a reference starting point, and it is assigned a value of zero. We omit the origin point in our graphical representation, instead, we use self-loop arrows to represent the earliest and latest start times, and earliest and latest finish times. These give the absolute time when tasks must be executed. Thus, start and finish time points must be executed within the intervals $[ES_t, LS_t]$ and $[ES_t + DUR_t, LS_t + DUR_t]$, respectively.

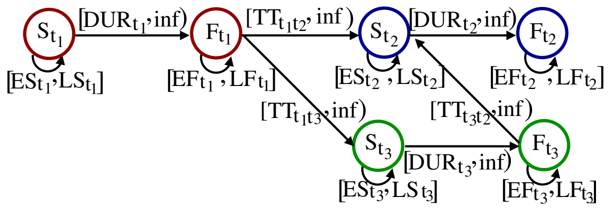


Figure 1: Example of an STN for the tasks assigned to a robot. The STN has the time points, duration and travel time constraints for three tasks assigned to the robot.

We consider two types of constraints between pairs of time points: duration and travel time constraints. Duration constraints are imposed if the time points belong to the same task. Travel time constraints are imposed when one time point is the end of a task and the other is the start of the next task. Both types of constraints induce an ordering of events. The start of a task cannot occur after its finish time ($F_t - S_t \in [DUR_t, \infty)$); and a robot reaches its next task t only after finishing its previous task t' ($S_t - F_{t'} \in [TT_{t',t}, \infty)$), where TT is the travel time between the two locations.

Algorithm 3 computes a bid for a task by trying to insert the task in each available time segment in the robot schedule and selecting the place that minimizes the makespan without causing temporal conflicts. The algorithm assumes that each robot keeps a working copy of its STN, which is used while bidding, and the final STN that holds the temporal information of the tasks is assigned to the robot.

When a new task t'' is to be added to a robot schedule, the start and finish time points for t'' are added to the working STN, along with the duration and travel time constraints. If t'' is the first task in a robot's schedule, we solve the STN by assigning $\max(TT_{r,t''}, ES_{t''})$ to the task's start time point ($S_{t''}$) if $S_{t''} \leq LS_{t''}$. If the robot's schedule contains other tasks, to schedule t'' the algorithm needs to find a place to fit t'' without making the STN inconsistent. When there are multiple places a task can be inserted, all insertion points are tried and the one that minimizes the makespan is returned.

The time points of a task are associated with two constraints, one between its start time and the finish time point of the previous task, and the other from its finish time point and the start time point of the next task in the schedule. Once these are added, the STN is propagated using the Floyd-Warshall algorithm. If no negative cycles occur, the network is consistent. The makespan of the schedule is computed by finding the STN solution with the smallest makespan.

Algorithm 3: Task Scheduling Algorithm

```
Input: Robot location $r_{loc}$, robot schedule $\bar{T}_r$, task to
       insert $t_{ins}$, STN for $r$.
Output: Index indicating the task insertion position in
        the robot schedule and the makespan that is
        used for bidding.
1  index $i = -1$;
2  if $\bar{T}_r =$ null then
3      $makespan = computeMakespan$;
4      return $0, makespan$;
5  else
6      for
       $i = 0, ..., m$ where m is the number of tasks in STN
       do
7          Insert $t_{ins}$ at position $i$ in $\bar{T}_r$;
8          Add task time points and all constraints to STN;
9          Propagate the STN using Floyd-Warshall;
10         if STN is consistent then
11             $makespan = computeMakespan$;
12             if makespan is smallest so far then
13                 save $i, makespan$;
14         reset STN to the copy prior to inserting task $i$;
15 if i=-1 then
16     return $-1, \infty$ ;
17 else
18     return $i, makespan$ ;
```

The STN is reset before trying each insertion point to ensure only the tasks allocated to the robot are included. The final STN is updated only when the robot wins a task.

Bidding Rules

We consider two team objectives: to minimize the makespan (equivalent to MiniMAX (Lagoudakis et al. 2005)) and to minimize a combination of makespan and distance traveled. Since we assume the robots have the same speed, distance is transformed into travel time. To prevent conflicting assignments all the robots use the same bidding rule.

Bid with Makespan. Robots bid using the makespan of their schedules computed with Algorithm 3.

Bid with Combined Makespan and Distance. The bid value for a task is a linear combination of the makespan and the distance traveled, as in (Solomon 1987): $TT_{t,\hat{t}} - \sum_{t,\hat{t}\in\bar{T}_r} TT_{t,\hat{t}}]$, where $\alpha \in [0, 1]$ is a weighting factor and M^r_{max} is the maximum makespan for r. In our experiments we set $\alpha = 0.5$. We call this bidding rule the dual objective heuristic (duo) and the TeSSI version that uses it TeSSIduo.

TeSSI Algorithm Analysis

TeSSI produces a total of $n \times m$ bids in each iteration, for a total of m iterations, producing an $O(nm^2)$ complexity. The scheduling algorithm requires at most $O(|\bar{T}_r|^2)$ to build the schedule for each robot. The worst-case complexity is $O(m^2)$, which occurs when one robot is assigned all the tasks. The complexity of propagation of the STN is the same as the complexity for the Floyd-Warshall algorithm, which is is $O(m^3)$ in the worst-case. The makespan computation is linear in the number of tasks. The algorithm simply walks through the vector of time points once. TeSSIduo adds $O(|\bar{T}_r|)$ extra distance computations to TeSSI.

The auctioneer uses n priority queues to compute the winners of the auction resulting in a $O(nlog_2m)$ complexity. The communication complexity is $O(nm)$ since the auction has m iterations and n robots send one message per iteration.

When tasks are not allocated, they are removed from the list of tasks to be auctioned, hence the list of tasks available for auctioning becomes empty after m iterations. TeSSI and TeSSIDuo always terminate, but are not complete and do not guarantee an optimal solution, due to the use of sequential single item auctions. The number of robots may need to be increased to allocate all the tasks. This practice is common in the vehicle routing (VRP) literature (Toth and Vigo 2002).

TeSSI Auction Example

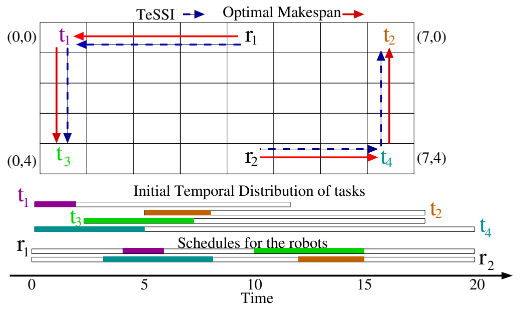


Figure 2: Allocations for the example scenario.

Here we give a brief example on how the algorithm works. We use a simple scenario (see Figure 2) with two robots and four tasks. The time windows for each task are shown with rectangles, where the color shaded areas indicate task durations. Tasks durations are 2, 3, 5, 5 for tasks 1-4, respectively. The time windows are [0,12], [5,18], [2,18] and [0,20] for tasks 1-4, respectively. Tasks' locations are (0,0), (7,0), (0,4) and (7,4) for tasks 1-4, respectively. Robot locations are (4,0) for r_1 and (4,4) for r_2. We use Cartesian distances. We assume that r_1 and r_2 move 1 grid element per time unit.

In the first iteration, to bid on task t_1 robot r_1 computes the makespan if t_1 is added to its schedule as follows: $max(4, 0) + 2 = 6$, where 4 is the travel time from r_1 to t_1, 0 is the earliest start time for t_1 and 2 is t_1's duration. Observe that the $max(4, 0) < 10$, where 10 is the latest start time for t_1 which is computed from its latest finish time as ($LS_{t_1} = LF_{t_1} - DUR_{t_1}$). Hence the robot can reach the task and perform it without violating t_1's temporal constraints. The bids for the other tasks are computed in a similar way. The bid vectors in the first iteration are for r_1 $[(t_1, \mathbf{6}), (t_2, 8), (t_3, 10.7), (t_4, 10)]$ and for r_2 $[(t_1, 7.7), (t_2, 8), (t_3, 9), (t_4, 8)]$. Tasks t_1 has the minimum bid, the auctioneer picks t_1 and allocates it to r_1, then removes t_1 from the task list.

In the next iteration, r_2 bids remain the same for the remaining tasks (t_2, t_3, t_4) because it did not win any tasks in the previous iteration. However, the bids for r_1 change. For example, when bidding on t_2, r_1 attempts to insert t_2 both before and after t_1. If t_2 is inserted before t_1 the computation returns 17 (3 travel to t_2 + 2 wait for early start of t_2 + 3 duration of t_2's + 7 travel to t_1 + 2 duration of t_1). If instead t_2 is inserted after t_1, the computation returns 16 (6 previous bid for t_1 + 7 travel time from t_1 to t_2 + 3 duration of t_2's). Hence the bid where t_2 is after t_1 has a lower makespan and it is the one submitted for t_2. The remaining bids are computed in the same way.

The overall makespan TeSSI returns (15) is the same as the optimal makespan. TeSSIduo returns the same allocation. Note, however, that TeSSI and TeSSIDuo do not guarantee an optimal allocation due to the fact that sequential single item auctions do not make such guarantee.

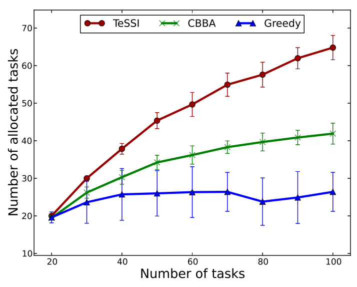
TeSSI Experiments and Results

We evaluate our algorithm in simulation in three experiments. In the first, all the tasks are known upfront, while in the second the tasks arrive dynamically. The third experiment uses a dataset from the VRP with time windows literature. We compare the performance of our algorithm against a version of CBBA (Ponda et al. 2010) that handles time windows and a greedy algorithm. The greedy algorithm iterates through the list of tasks once, and for each task iterates through the list of robots searching for the robot which can do that task and attain the lowest makespan. Each robot uses Algorithm 3 on each single task to compute its makespan.

Data Generation

Experiment 1. We generated two datasets according to the random task generator proposed by Ponda et al. (2010) on a 60 by 60 2D grid. The tasks in both datasets have a fixed duration (15 time units). The time windows are randomly generated with earliest start times uniformly drawn from [0,100]. Latest start times are computed by adding to the earliest start times a random number drawn from the range [1, duration of the task]. The tasks' locations in datasets 1 and dataset 2 are generated uniformly over the 2D grid. In dataset 1 there are 20-100 tasks (in 10 task increments), and 10 robots. In dataset 2, there are 100 tasks and a variable number of robots (5 to 40 in 5 robot increments, and 50). The datasets have varying degrees of difficulty with respect to the tightness of the time windows.

Experiment 2. For the dynamic task arrival case, we used the data for 100 tasks and 10 robots from Experiment 1. For this simulation, we used the JADE (Bellifemine, Poggi, and Rimassa 1999) multi-agent platform. Each robot is a JADE agent, and there is an auctioneer agent. In this setup, tasks arrive in batches of sizes 1, 5, 10, and increments of 10 up



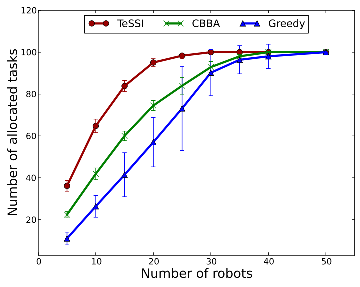


Figure 3: Experiment 1: Number of tasks allocated (left) for dataset 1 (having 20-100 tasks with 10 robots), and (right) for dataset 2 (having 100 tasks with 5-50 robots).

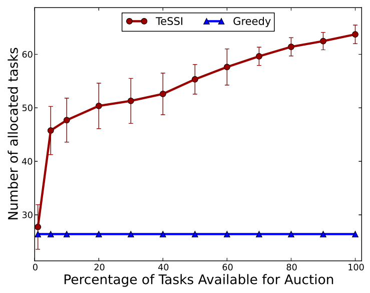


Figure 4: Experiment 2: Number of tasks allocated to 10 robots. 100 tasks arrive dynamically in batches of 1,5,10-100.

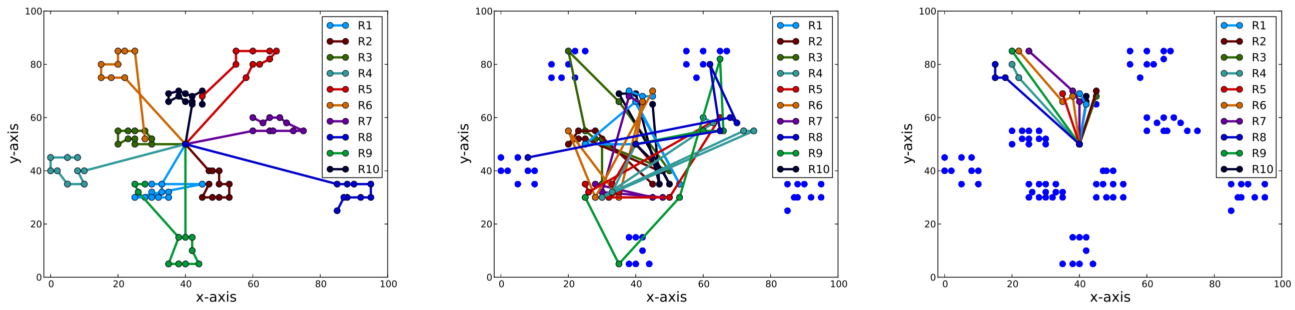


Figure 5: Experiment 3: Allocations done by TeSSI (left), CBBA (mid) and Greedy (right) on Solomon's instance C101.

to 100. Tasks are added at the beginning of each iteration of the auction algorithm. An auction iteration ends when the list of tasks available for bidding is empty. If the system is idle, due to the lack of available tasks, a new arrival triggers another auction iteration.

Experiment 3. We used the spatial and temporal data in the Solomon's dataset (Solomon 1987), a standard dataset used for VRP with time windows. To simplify our analysis we do not use the capacity and scheduling horizon data. The dataset includes six types of problems, with different locations: R (distributed randomly), C (clustered), and RC (mixed random and clustered) problem types, which refer to different spatial distributions of tasks. For each of the R, C and RC types there are two types of datasets depending on the time windows. Type 1 datasets have narrow time-windows, while type 2 datasets have large time-windows. Each of the six datasets, R1, C1, RC1, R2, C2 and RC2, contains 8–12 different time-window and task location configurations. Each configuration includes 100 tasks and 10 robots.

The algorithms were run on 30 instances for each number of tasks in dataset 1 and each number of robots in dataset 2 for Experiment 1, 30 for Experiment 2, and 8-12 instances for Experiment 3. For Experiments 1 and 2 both bidding rules for TeSSI perform similarly. Hence, we only report results when robots bid using makespan. We report the total number of tasks allocated in Experiments 1 and 2 (Figures 3 and 4), and, in addition, the makespan and distance information for Experiment 3 (Table 1).

Results and Analysis

In comparing TeSSI to CBBA and the greedy algorithm, we found that TeSSI outperforms both in terms of the number of tasks allocated and the number of robots used. TeSSI allocates more tasks on average than CBBA and the greedy algorithm for both datasets in Experiment 1 (see Figure 3).

We found that TeSSI uses fewer robots to allocate the same number of tasks compared to the other algorithms when using dataset 2 in Experiment 1. TeSSI uses 30 robots to completely allocate the 100 tasks, while CBBA and the greedy algorithm use 40 and 50, respectively. We also found that the makespan TeSSI obtains is competitive with CBBA (both with average makespan=114) when both algorithms are able to allocate all of the tasks. This is shown in Experiment 1 using 40 robots and 100 tasks in dataset 2. We do not report further comparisons of makespan and the distance covered by the robots for Experiment 1, since this is relevant only when the algorithms allocate the same number of tasks.

TeSSI allocates more tasks than CBBA because in the makespan minimization process it allows the already allocated tasks to move around to accommodate the insertion of new tasks. This enables robots to pack more tasks into their schedules. Conversely, CBBA allocates fewer tasks because it needs to obey the principle of diminishing marginal gains. Since the scoring function used depends on the time a robot arrives at a task, CBBA assumes that the addition of a new task does not change the order nor the arrival times for tasks already allocated. However, this is very restrictive. For ex-

		TeSSI			TeSSIduo			CBBA			Greedy		
Data		Makespan	Distance	# Tasks	Makespan	Distance	# Tasks	Makespan	Distance	# Tasks	Makespan	Distance	# Tasks
R1	μ	201.75	915.07	82.33	204	838.39	82.33	182.25	1051.52	52.25	181.83	902.47	42.25
	σ	8.30	49.70	8.17	6.66	35.98	7.22	3.33	239.63	8.36	14.65	254.47	14.96
C1	μ	1115.89	1265.03	92.89	1110.00	1003.31	96.22	1039.11	1351.50	57.89	1041.67	1234.09	56.22
	σ	49.83	315.06	5.44	56.77	373.77	4.52	90.63	201.08	11.34	66.05	597.82	22.65
RC1	μ	207.13	948.33	100.00	204.13	843.81	100.00	182.00	1027.13	47.25	164.75	750.51	25.13
	σ	31.20	53.65	0.00	7.74	37.43	0.00	31.20	156.93	7.96	14.81	150.49	3.31
R2	μ	774.00	2218.59	100.00	775.64	1338.69	100.00	772.36	2171.70	75.91	773.82	2562.36	100.00
	σ	106.72	367.95	0.00	104.98	122.17	0.00	111.22	754.41	23.30	107.221	217.93	0.00
C2	μ	3088.88	1954.31	100.00	3093.75	1081.95	100.00	3088.88	1820.70	82.62	3088.88	2289.87	100.00
	σ	157.60	891.93	0.00	162.38	369.83	0.00	157.60	805.70	24.58	157.60	204.96	0.00
RC2	μ	759.00	2859.58	100.00	761.00	1493.56	100.00	753.38	2522.26	80.75	757.25	2981.72	100.00
	σ	126.32	384.23	0.00	126.67	139.65	0.00	139.65	844.40	23.43	130.90	240.47	0.00

Table 1: Experiment 3: Mean (μ) and standard deviation (σ) values for makespan, total distance traveled, and number of tasks allocated obtained by TeSSI (both bidding rules), CBBA, and Greedy using Solomon's data instances.

ample, suppose a vehicle first gets a task with a large duration and a large time window. If it sets its arrival time at the start of the time window, then any smaller task with a tight window that the robot could have done before the long task will not be allocated to that robot. This is a missed opportunity. Our work overcomes this by using bidding functions that rely on makespan instead of individual arrival times.

In Experiment 2, where tasks arrive dynamically, the performance of TeSSI is the same as for the greedy algorithm (see Figure 4) in cases when the number of tasks available is too low to leverage the synergies among tasks. However, the performance of the algorithm improves quickly when 5 or more (out of 100) tasks arrive in each iteration of the auction. The improvement is produced by the availability of more tasks at once, which increases the ability of the robots to produce a higher quality solution. This shows that the algorithm does not require large numbers of tasks to improve its allocations. This is advantageous in real life situations where tasks may not arrive in large batches.

In Experiment 3, TeSSI and TeSSIduo allocate more tasks when task locations are clustered than when they are randomly distributed (Solomon's type 1 data in Table 1). The main reason is that robots are distributed to different clusters of tasks, and travel less often to tasks outside their clusters. This is evidenced in Figure 5 where in many cases each robot route for TeSSI visits only one cluster (same for TeSSIduo). The other algorithms do not take full advantage of the spatial relationships among tasks and produce allocations that result in lower numbers of tasks allocated.

The makespan and number of tasks allocated are very similar for both bidding heuristics for TeSSI when using Solomon's type 2 data in Experiment 3. However, the dual objective heuristic consistently generates paths of lower total length than the heuristic that bids only with makespan (see Table 1). This emphasizes the advantage of using distance (or travel time) when computing bids. TeSSIduo outperforms all the other methods when distances are considered, and it does so without dramatically increasing the makespan.

Moreover, TeSSI, TeSSIduo and the greedy algorithm are able to allocate all the tasks with similar makespans for the type 2 data. This is because the temporal constraints in these datasets are much looser than the ones for type 1 data. The greedy algorithm, normally the worst performing method, is still able to allocate all tasks because it takes advantage of the type 2 problems' looser constraints and moves tasks around in a robot's schedule.

TeSSI's computation time is two orders of magnitude smaller than that of CBBA in all datasets for Experiment 3, and is competitive with that of the greedy algorithm. For example, it takes CBBA an average of 98.9 seconds when allocating 100 tasks to 10 robots using C2 data, while it takes TeSSI only 0.43 seconds on the same dataset. We also found that TeSSI (same for TeSSIduo) spends twice the amount of time when using type 2 data than when using type 1 data for Experiment 3. The main reason is that the larger temporal flexibility in type 2 generates more insertion points for new tasks and the algorithm tries all insertion points to find the one that minimizes the makespan of a robot's schedule.

Overall, if the total distance robots travel is a concern (for example, robots working in large areas), TeSSIDuo is a clear choice. If time to finish is more important than distance (for example, robots working in small areas), then TeSSI is a better choice because on average it is twice as fast as TeSSIDuo.

Conclusions and Future Work

We presented the TeSSI algorithm, which allocates tasks with time windows to cooperative robots. The main features of the algorithm are a fast and systematic processing of temporal constraints and two bidding methods that optimize either completion time, or a combination of completion time and distance. Results show that TeSSI's computation times are efficient, and that it completes more tasks than other methods on randomly generated data and on datasets used for VRP. Future work involves studying task execution uncertainties, introducing auction and schedule repair mechanisms, which we hope will further improve the performance, and allowing use in more dynamic environments.

Acknowledgment: Partial support is gratefully acknowledged from NSF grant IIS-1208413.

References

Alighanbari, M.; Kuwata, Y.; and How, J. P. 2003. Coordination and control of multiple UAVs with timing constraints and loitering. In *American Control Conf.*, 5311–5316.

Bellifemine, F.; Poggi, A.; and Rimassa, G. 1999. *JADE - A FIPA-compliant agent framework*. The Practical Application Company Ltd. 97–108.

Choi, H.-L.; Brunet, L.; and How, J. 2009. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans. on Robotics* 25(4):912 –926.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence Journal* 49(1-3):61–95.

Dias, M. B.; Zlot, R.; Kalra, N.; and Stentz, A. 2006. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE* 94(7):1257–1270.

Gombolay, M.; Wilcox, R.; and Shah, J. 2013. Fast scheduling of multi-robot teams with temporospatial constraints. In *Robotics: Science and Systems (RSS)*.

Koenig, S.; Tovey, C.; Lagoudakis, M.; Markakis, V.; Kempe, D.; Keskinocak, P.; Kleywegt, A.; Meyerson, A.; and Jain, S. 2006. The power of sequential single-item auctions for agent coordination. In *Proc. Nat'l Conf. on Artificial Intelligence*, 1625–1629.

Koenig, S.; Keskinocak, P.; and Tovey, C. A. 2010. Progress on agent coordination with cooperative auctions. In *Proc. Nat'l Conf. on Artificial Intelligence*, 1713–1717.

Korsah, G. A.; Kannan, B.; Fanaswala, I. A.; and Dias, M. B. 2010. Enhancing market-based task allocation with optimal seed schedules. In *Proc. of the Int'l Conf. on Intelligent Autonomous Systems*, 249 – 258.

Kumar, T. S.; Cirillo, M.; and Koenig, S. 2013. On the traveling salesman problem with simple temporal constraints. In *Proc. the 10th Symposium on Abstraction, Reformulation, and Approximation (SARA)*.

Lagoudakis, M. G.; Markakis, E.; Kempe, D.; Keskinocak, P.; Kleywegt, A.; Koenig, S.; Tovey, C.; Meyerson, A.; and Jain, S. 2005. Auction-based multi-robot routing. In *Robotics: Science and Systems (RSS)*, 343–350.

Melvin, J.; Keskinocak, P.; Koenig, S.; Tovey, C. A.; and Ozkaya, B. Y. 2007. Multi-robot routing with rewards and disjoint time windows. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2332–2337.

Nanjanath, M., and Gini, M. 2010. Repeated auctions for robust task execution by a robot team. *Robotics and Autonomous Systems* 58(7):900–909.

Nunes, E.; Nanjanath, M.; and Gini, M. 2012. Auctioning robotic tasks with overlapping time windows. In *Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, 1211–1212.

Ponda, S. S.; Redding, J.; Choi, H.-L.; How, J.; Vavrina, M.; and Vian, J. 2010. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conf.*, 3998 –4003.

Solomon, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35(2):254–265.

Toth, P., and Vigo, D., eds. 2002. *The vehicle routing problem*. Philadelphia, PA: SIAM Monographs on Discrete Mathematics and Applications.