Repeated Sequential Auctions with Dynamic Task Clusters

Bradford Heap and **Maurice Pagnucco**
ARC Centre of Excellence in Autonomous Systems
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW, 2052, Australia
{bradfordh,morri}@cse.unsw.edu.au

Abstract

Sequential auctions can be used to provide solutions to the multi-robot task-allocation problem. In this paper we extend previous work on sequential auctions and propose an algorithm that clusters and auctions uninitiated task clusters repeatedly upon the completion of individual tasks. We demonstrate empirically that our algorithm results in lower overall team costs than other sequential auction algorithms that only assign tasks once.

Introduction

We consider the problem of a team of autonomous mobile robots operating in a known office-like environment. These robots may be required to deliver documents between offices, clean up spillages, or act as tour guides to visitors. In these situations there is a set of tasks to be completed and we require the robots to distribute these tasks amongst themselves in a manner that satisfies a global team objective.

Market-based approaches to distributed task-allocation have received significant research attention in recent years (Dias et al. 2006; Koenig, Keskinocak, and Tovey 2010). An optimal allocation of a set of tasks to robots can be determined by a *single-round combinatorial auction*. However, this auction algorithm is NP-Complete and suffers from high communication and winner determination costs (Berhault et al. 2003). As an alternative, *sequential single-item auctions* (SSI auctions) which allocate tasks over multiple rounds have been well studied (Lagoudakis et al. 2005; Koenig et al. 2006). Despite SSI auctions producing team costs that are generally sub-optimal, they have much lower communication and winner determination costs which result in a much faster allocation of tasks. Various improvements and extensions to SSI auctions have been suggested which trade off allocation time against overall team costs (Koenig, Keskinocak, and Tovey 2010).

This paper combines and extends the ideas presented in two previous works on SSI auction extensions: sequential auctioning of task clusters (Heap and Pagnucco 2011) and repeated auctions of uncompleted tasks (Nanjanath and Gini 2010). This previous work has shown that when clusters of tasks that employ positive inter-task synergies are auctioned in lieu of single tasks, the resultant overall team costs can be lower. Meanwhile work on repeated auctions of uncompleted individual tasks upon the completion of a single task has shown the robustness of robotic teams in reallocating tasks when unexpected delays occur while completing tasks (Nanjanath and Gini 2010). Our idea is, that upon completion of a single task, all robots create clusters of their uninitiated tasks and auction these task clusters with the goal of improving the minimisation of the overall team objective.

In the remainder of this paper we define the task-allocation problem in the domain of auction-like algorithms, we look at related work on task clustering and post-initial allocation improvements to task-allocation, we describe our algorithm, provide complexity analysis, and report empirical results obtained in simulation of robots in an office-like environment. Our key empirical result shows a maximum cost improvement of 4.8% compared to task allocation using SSI auctions and in general the greater the number of robots and tasks the larger the improvement.

Multi-robot Task-Allocation

We now formalise the definition of the task-allocation problem in a similar manner to (Koenig et al. 2007). Given a set of robots $R = \{r_1, \ldots, r_m\}$ and a set of tasks $T = \{t_1, \ldots, t_n\}$, any tuple $\langle T_{r_1}, \ldots, T_{r_m} \rangle$ of pairwise disjoint bundles $T_{r_i} \subseteq T$ and $T_{r_i} \cap T_{r_j} = \emptyset$ for $i \neq j$, for all $i = 1, \ldots, m$, is a partial solution of the task-allocation problem. This means that robot r_i performs the tasks T_{r_i}, and no task is assigned to more than one robot. To determine a complete solution to the task-allocation problem we need to find a partial solution $\langle T_{r_1} \ldots T_{r_m} \rangle$ with $\cup_{r_i \in R} T_{r_i} = T$, that is, where every task is assigned to exactly one robot.

Multi-robot routing is considered the standard testbed for the task-allocation problem (Dias et al. 2006). The tasks represent locations to visit. Robots know their locations and can calculate the costs to travel between locations. We assume costs are symmetric, $\lambda(i, j) = \lambda(j, i)$ and are the same for all robots. The robot cost $\lambda_r(T_r)$ is the minimum cost for an individual robot r to visit all locations T_r assigned to it. There can be positive synergies between two tasks where $\lambda_{r_i}(T_{r'} \cup T_{r''}) < \lambda_{r_i}(T_{r'}) + \lambda_{r_i}(T_{r''})$. Robots can also have capacity constraints where they have a fixed maximum number of locations to visit. We wish to find a solution to the task-allocation problem that achieves a team objective. In

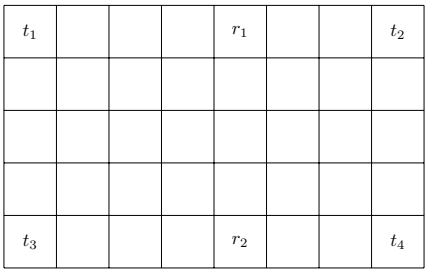

Figure 1: Exploration Task 4 (Koenig et al. 2006).

this paper we use two common team objectives introduced in (Tovey et al. 2005):

MiniMax $\max_{r \in R} \lambda_r(T_r)$, that is to minimise the maximum distance each individual robot travels.

MiniSum $\sum_{r \in R} \lambda_r(T_r)$, that is to minimise the sum of the paths of all robots in visiting all their assigned locations.

Although SSI auctions guarantee a solution to the task-allocation problem they do not guarantee that this solution will be optimal for the team objective. Despite this, (Lagoudakis et al. 2005) provides us with theoretical guarantees on the bounds of the solution in the domain of multi-robot routing which is the standard testbed of the auction-based coordination systems.

One problem with standard SSI auctions is when robots have few allocated tasks they fail to consider many inter-task synergies when calculating bids. As a result, robots end up with a bias towards tasks that are nearby; this is particularly apparent during early bidding rounds. An example of this can be seen in Exploration Task 4 (Koenig et al. 2006) (Figure 1). If we apply a normal SSI auction to this example, during the first two bidding rounds t_2 is allocated to r_1 and t_4 is allocated to r_2. During the next two bidding rounds t_1 is allocated to r_1 and t_3 is allocated to r_2. This solution is sub-optimal.

To avoid this initial bias (Heap and Pagnucco 2011) proposed the formation of task clusters prior to the auction process. Robots then bid on task clusters rather than single items. In their empirical experiments they apply *K*-means clustering to form task clusters and their results show that, on average, sequential auctions with clusters result in lower overall team costs with a similar runtime to SSI auctions. Applying this approach to the same example, we form two clusters $c_1 = \{t_1, t_3\}$ and $c_2 = \{t_2, t_4\}$. The first round of bidding assigns c_2 to r_1, and the second round assigns c_1 to r_2. This allocation of tasks to robots is the optimal solution.

Another recent auction algorithm using *K*-means clustering has been presented by (Elango, Nachiappan, and Tiwari 2011), however, this technique differs from the goals of sequential-class auctions and instead seeks to evenly balance tasks across all robots. Furthermore, *K*-means clustering has also been used for coordinating multi-robot exploration (Solanas and Garcia 2004). This research used clustering to partition unexplored space into a number of disjoint regions—fixed to the number of robots. After clustering, robots are centrally assigned the disjoint region they are closest to. Through the partitioning of space in this manner robots are quickly dispersed throughout the entire enviornment. Recent analysis (Puig, Garcia, and Wu 2011) compared this approach to other state-of-the-art approaches and demonstrated its effectiveness.

1 **function SSC-Auction** ($\bar{C}$,C_r, r, R)
2 **Input:** $\bar{C}$: the set of clusters to be assigned C
3 C_r: the set of clusters presently assigned
4 r: the robot r
5 R: the set of robots R
6 **Output:** C_r: the set of clusters assigned to the robot
7 **while** ($\bar{C} \neq \emptyset$) **do**
8 */* Bidding Stage */*
9 **for** each cluster $c \in \bar{C}$ **do**
10 $b_r^c \leftarrow$CalculateBid(C_r,c);
Send(b_r^c, R) | $B \leftarrow \bigcup_i$Recieve($b_{r_i}^c$, R);
11
12 */* Winner-Determination Stage */*
13 $(r', c) \leftarrow \arg\min_{(r' \in R, c \in \bar{C})} B$;
14 **if** $r = r'$ **then**
15 $C_r \leftarrow C_r \cup \{c\}$;
16 $\bar{C} \leftarrow \bar{C} \backslash \{c\}$;

Figure 2: Sequential Single-Cluster Auctions.

To further refine a task-allocation solution post-initial allocation (Nanjanath and Gini 2010) present an algorithm for the repeated auctioning of all uncompleted tasks upon the completion of each task. In this algorithm, robots will only exchange tasks if it improves the overall team objective. Once a task has been exchanged, the robots involved in the exchange re-plan their paths to minimise the total distance travelled. Their empirical results show that the final task-allocation is close to optimal.

An alternative approach to re-allocating tasks is through task exchanges. A common approach to this is repeated single task exchanges (Dias and Stentz 2000). A recent extension of this is *K-Swaps* (Zheng and Koenig 2009) which exchanges many tasks between multiple robots at the same time. K-Swaps has been experimentally shown to make significant improvements to task-allocations, however, the algorithm trades off larger improvements with pronouncedly increased computational time.

Sequential Single-Cluster (SSC) Auctions

Sequential Single-Cluster (SSC) auctions (Heap and Pagnucco 2011) are an extension of SSI auctions and assign clusters of tasks to robots in multiple bidding rounds. At the conclusion of each bidding round one previously unassigned task cluster is awarded to the robot that bids the least for it so that the team cost increases the least. Once all task clusters are allocated all robots complete all tasks allocated to them in as short a distance possible. Robots do not have to do all tasks in a cluster sequentially. After a cluster has been awarded, the tasks within the awarded cluster can be reordered with tasks from previously allocated clusters.

We formulate the algorithm for SSC auctions in Figure 2.

Each robot runs the algorithm independently of other robots and, with the exception of supplying the initial list of tasks and clusters to each robot, there is no centralised controller. Before the SSC auction algorithm begins, a clustering algorithm is used to allocate all individual tasks into task clusters with the goal of maximising the positive synergy between tasks in each cluster. Each task is assigned to one, and only one cluster, and clusters can be of varying sizes. All robots are then informed of all tasks and all clusters.

The SSC auction begins and continues while there are unassigned task clusters (Line 7). The bidding stage (Lines 8-11) consists of the robot calculating bids for every unassigned task cluster and submitting these bids to all other robots. Each bid calculation requires robots to provide a solution to the travelling repairman problem (Blum et al. 1994). Because this problem is NP-hard, robots often use the cheapest-insertion and two-opt heuristics (Croes 1958) to provide a close approximation to the optimal solution. Each bid is a triple of a robot b_r, a task cluster b_c and a bid cost b_λ, such that, $b = \langle b_r, b_c, b_\lambda \rangle$. The function CalculateBid takes the set of previously assigned clusters C_r and the cluster c being bid on and uses a bidding rule to calculate a bid cost (Line 10). The robots send their bids and receive all bids from other robots in parallel (Line 11). The winner-determination stage (Lines 12-16) consists of each robot choosing the task cluster with the lowest bid from the set of submitted bids. Ties can be broken in an arbitrary way. The robot with the winning bid has the winning task cluster assigned to it. All robots then remove the winning task cluster from the set of unassigned clusters and the next bidding round begins.

Repeated Auctions with Dynamic Clustering

We now describe our procedure for repeated auctions of task clusters upon individual task completion. Our algorithm assumes a homogenous set of robots which are supplied with a map of the enviornment, have perfect localisation, have error free communication with other robots, and do not break down. Auctions are run sequentially, that is, if an auction is running after the completion of a task and a second robot completes a task, the auction for the second task completion does not begin until after the first auction is complete. We also make the assumption that auctions complete as quickly as possible and the time to complete tasks is much greater than the time to run an auction to reallocate tasks.

Our alogrithm operates as follows: when a robot completes a task it signals to all other robots that an auction for the redistribution of tasks is to begin. All robots then create clusters of uncompleted tasks, and inform all other robots of these clusters, excluding the cluster containing the task they are currently completing. The robots then all run a SSC-Auction on these clusters. After all clusters have been distributed via the auction each robot replans its path based on its new allocation of tasks. While the auction runs each robot continues to complete its current task.

The algorithm that each robot executes is presented in Figure 3. All robots are assigned an initial allocation of tasks and a clustering factor which is used to calculate K for clustering tasks. The algorithm runs on all robots until all tasks

```
1  function RepeatedSSCAuctions (T, r, R, CF)
2  Input: T: the set of Tasks to be completed T
3         r: the robot r
4         R: the set of robots R
5         CF: the factor of clusters to tasks
6  Output: T = ∅: All tasks completed
7  P_{T_r} ← MinimisePath(T_r);
8  while T ≠ ∅ do
9      if T_r ≠ ∅ then
10         DriveToTask(t_i ∈ P_{T_r}, r, R) |
           T_r ←ListenForAuction(t_i ∈ P_{T_r}, T_r, R, CF);
11     else
12         T_r ←ListenForAuction(t_i ∈ P_{T_r}, T_r, R, CF);
13 function DriveToTask (t, r, R)
14 Input: t: the task to be completed t
15        r: the robot r to drive
16        R: the set of robots R
17 Output: r_l = t_l: The robot r located at task t
18 while r_l ≠ t_l do
19     Robot r moves towards task t
20 Signal all robots r_i ∈ R to begin auction;
21 Wait for auction to finish;
22 T_r ← T_r \ {t_i};
23 P_{T_r} ← MinimisePath(T_r);
24 function ListenForAuction (t, T_r, r, R, CF)
25 Input: t: the currently initialised task t
26        T_r: the set of Tasks allocated to the robot T_r
27        r: the robot r
28        R: the set of robots R
29        CF: the factor of clusters to tasks
30 Output: T_r: the set of Tasks allocated to the robot T_r
31 Wait for signal to begin auction;
32 C_r ← KMeansClustering(K ← CF * |T_r|, T_r);
33 C̄_r ← C_r \ {c_t ∈ C_r};
   Send(C̄_r, R) | C̄ ← ⋃_i Recieve(C_{r_i}, R);
34
35 if C̄ ≠ ∅ then
36     C_r ← SSC-Auction(C̄, {c_t ∈ C_r}, r, R);
37     T_r ← {t ∈ c | c ∈ C_r} ;
38 if r_l = t_l then
39     Signal Auction Finished;
40 else if T_r ≠ ∅ then
41     ListenForAuction(t, T_r, r, R, CF);
```

Figure 3: Repeated Auctions with Dynamic Clustering.

are completed. Initially each robot plans a path to complete all allocated tasks (Line 7). Each robot then drives to the first task in its path and in parallel listens for a signal to begin an auction (Line 10). However, if a robot has no tasks currently allocated to it then it just listens for the signal to begin an auction (Line 12).

The function DriveToTask (Lines 13-23) controls the movement of the robot towards its current task and signals the start of an auction upon arrival at the task. The robot will continue travelling towards its current task until it reaches it

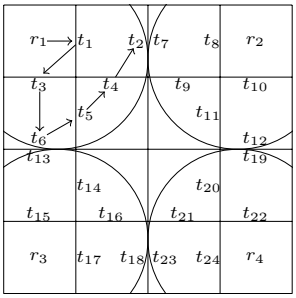

Figure 4: Initial Allocations.

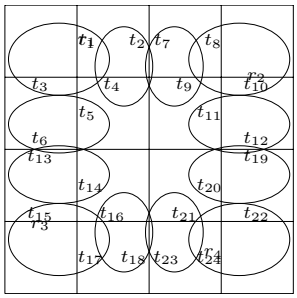

Figure 5: Cluster Formation.

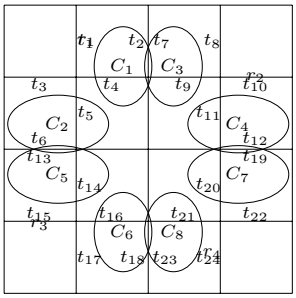

Figure 6: Cluster Auction.

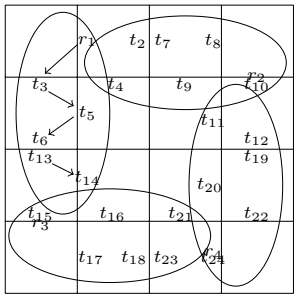

Figure 7: New Allocation.

(Lines 18-19). Once the robot arrives at its current task it will signal all robots to begin an auction and wait until the auction is complete (Lines 20-21). Upon completion of the auction the robot will remove its current task from its set of tasks to complete, replan its path (Lines 22-23). The function then terminates and if the robot still has tasks to complete it will begin travelling to the next task.

The function ListenForAuction (Lines 24-41) sets up and controls the SSC auction for reallocating the assigned tasks. Upon the signalling of an auction each robot forms clusters of tasks using K-means clustering (Line 32). To determine K in each auction we multiply the constant cluster factor CF by the number of currently allocated tasks. The constant cluster factor is a value between 0 and 1 and describes the ratio of tasks to clusters. In our experiments we typically use values of $\frac{1}{2}$ and $\frac{2}{3}$ for CF. This gives us an average of 2 tasks/cluster and 1.5 tasks/cluster respectively. After the formation of the task clusters the robot removes the cluster containing its currently initialised task (Line 33). This subset forms the robot's contribution to the set of clusters for auction which is sent to all robots. In parallel the robot recieves a set of clusters from all other robots. These sets are merged to form the complete set of all clusters for auction (Line 34). If there are clusters for auction, all robots run the SSC auction algorithm simultaneously (Line 36). The robot then resets its allocated tasks with all tasks in the post-auction cluster set (Line 37). Finally, the robot that signalled the start of the auction signals the end of the auction (Line 39) and other robots with tasks to complete listen for the next auction (Line 41).

Figures 4, 5, 6, and 7 provide an example of an auction using our algorithm with the MiniMax team objective. We have four robots and 24 tasks. Each robot has an initial allocation of tasks such that: $T_{r_1} = \{t_1, \ldots, t_6\}$, $T_{r_2} = \{t_7, \ldots, t_{12}\}$, $T_{r_3} = \{t_{13}, \ldots, t_{18}\}$ and $T_{r_4} = \{t_{19}, \ldots, t_{24}\}$ (Figure 4). Robot r_1 signals the start of an auction after arriving at task t_1. At this point r_2 is approaching t_{10}, r_3 is approaching t_{15}, and r_4 is approaching t_{24}. Each robot with $CF = \frac{1}{2}$ then uses *K*-means to allocate its tasks into three task clusters (Figure 5). All robots then remove the cluster containing their currently initialised task and exchange their remaining clusters with all robots to form the clusters for auction $\bar{C} = \{C_1, \ldots, C_8\}$. The SSC auction then begins (Figure 6). During the first four bidding rounds C_2 is allocated to r_1, C_3 is allocated to r_2, C_6 is allocated to r_3, and C_7 is allocated to r_4. The next four bidding rounds allocate the remaing clusters. C_5 is allocated to r_1, C_1 is allocated to r_2, C_8 is allocated to r_3, and C_4 is allocated to r_4. The complete new allocation is shown in Figure 6.

Algorithm Analysis

We now consider the communication requirements and expected results of repeated SSC auctions with dynamic clusters and compare this to standard SSI and SSC auctions.

For a distributed SSI auction there are $|R|$ robots and $|T|$ tasks and all robots are aware of all tasks before the auction. In each round of the auction every robot has to communicate its bid to every other robot so there are $|R|^2$ messages per round. In total there are $|T|$ rounds as only one task is allocated per round. Therefore in total there are $|T||R|^2$ messages sent in a complete auction.

SSC auctions follow the same bidding and communication rules as SSI auctions except that robots bid on clusters rather than tasks. All robots are aware of all tasks and clusters before the auction. Therefore the number of rounds in the auction is determined by the number of clusters $|C|$. Therefore it holds that in total there are $|C||R|^2$ messages sent in a complete auction. We also note that $|C| \leq |T|$ so generally there are less rounds and less communication in a SSC auction than in a SSI auction.

For repeated SSC auctions with dynamic clusters the number of messages exchanged in each round remains the same $|R|^2$. However, at the beginning of each auction all robots have to exchange their clusters to be auctioned, so this is an additional $|R|^2$ number of messages exchanged. There are then $|C|$ rounds per auction. Therefore in total there are $(1 + |C|) * |R|^2$ messages per auction and there are at most $|T|$ auctions one for each task completion. We can conclude that for repeated SSC auctions with dynamic clusters there are much greater communication requirements than auctions that only assign tasks once.

We now consider the expected results of each auction algorithm. (Koenig et al. 2006) show that for SSI auctions solving the multi-robot routing problem the strong lower bound is a factor of 1.5 away from the optimal and the upper bound is a factor of 2 away from the optimal.

For SSC auctions we can achieve an optimal solution in one auction as described earlier for the example scenario in Figure 1. However, the worst-case for SSC auctions is much greater than SSI auctions because there is no requirement

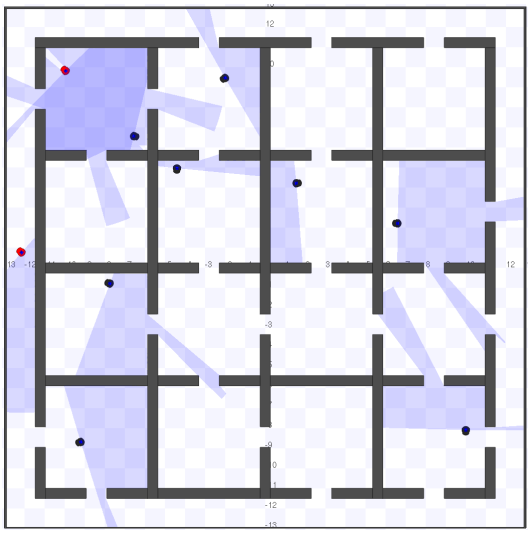

Figure 8: Simulation of an office-like environment.

that tasks assigned to clusters obey the triangle inequality. This means that two clusters can contain tasks that overlap each other but are assigned to different robots, resulting in a total path length that could be much less if all tasks were completed by only one robot. However, we conjecture that this worst-case situation would result in an allocation of tasks to robots that is no worse than a completely random allocation. Furthermore, with repeated auctions and dynamic clusters these worst-case allocations would most likely be redistributed and replanned before being executed.

Experiments

To test our algorithm we simulate an office-like environment with 16 rooms each containing four interconnecting doors that can be independently opened or closed to allow or restrict travel between rooms (Figure 8). This enviornment has become the de facto standard testbed in recent literature (Koenig et al. 2007; Zheng and Koenig 2009; Nanjanath and Gini 2010; Heap and Pagnucco 2011) and therefore it provides a common setting for comparison. In each experiment, the doors between different rooms and the hallway are either open or closed. We test on 25 randomly generated configurations of opened and closed doors with each robot starting in a different random location. Robots can only travel between rooms through open doors and they cannot open or close doors. However, it is guaranteed there is at least one path between each room and every other room. In each experiment robots are set a fixed maximum total task fulfilment capacity constraint of the double the number of tasks divided by the number of robots ($2 * \frac{|T|}{|R|}$). Robots stop being allocated additional tasks once these capacities are met. For each configuration we test with $|R| \in \{4, 6, 8, 10\}$, $|T| \in \{16, \ldots, 60\}$, and provide results for both the MiniSum and MiniMax team objectives.

Each experiment configuration is tested with three different initial task allocations: SSI auctions, SSC auctions with $|C| = \frac{1}{2}|T|$, and SSC auctions with $|C| = \frac{2}{3}|T|$. All three approaches produce a differing task to robot allocation and therefore different initial paths and costs. These initial path costs are included in our results (Tables 1 and 2) and provide reference for how much repeated auctions improve the overall team objective. We also test each experiment configuration with two different clustering factors for calculating K, $CF = \frac{1}{2}$ and $CF = \frac{2}{3}$.

Tables 1 and 2 show the mean results of our experiments with both the MiniSum and MiniMax team objectives respectively. In all robot/task combinations for the MiniSum team objective there is a reduction in the final cost where SSI auctions are used to generate the initial cost. However, for the MiniMax team objective we generally do not see an improvement when SSI auctions are used for the initial task allocation. For initial task allocations generated with SSC auctions we see the largest improvement in the final cost compared to the initial cost, however, we note that these experiments begin with the highest initial costs. We also observe a general pattern in the results of both team objectives that, as the number of robots increases, the percentage improvement also increases. For the MiniMax team objective the majority of the overall lowest final costs for each robot/task combination occur when the $CF = \frac{2}{3}$ and these lowest final costs occur evenly across all initial allocation techniques. However, for the MiniSum team objective there is a roughly even split between the lowest cost occurring when the $CF = \frac{1}{2}$ and $CF = \frac{2}{3}$ and the vast majority of overall lowest final costs occur when the initial allocation is created with SSI auctions.

Conclusions

In this paper we have described a new algorithm for solving the multi-robot task allocation problem. Our algorithm uses repeated auctions to reallocate uncompleted tasks amongst robots as individual tasks are completed. We provided an analytical evalution of the communication and the solution bounds. The results of our empirical experiments demonstrate this algorithm results in a better final allocation of tasks to robots than algorithms that only allocate tasks once.

We are currently considering future work including using different clustering techniques such as self-organising maps, testing on more complex environments, and forming clusters of robots for bidding rather than individual robot bidding. Furthermore, we plan to use these algorithms within the RoboCup Rescue Simulation League.

References

Berhault, M.; Huang, H.; Keskinocak, P.; Koenig, S.; Elmaghraby, W.; Griffin, P.; and Kleywegt, A. 2003. Robot exploration with combinatorial auctions. In *Proc. IROS-03*, 1957–1962.

Blum, A.; Chalasani, P.; Coppersmith, D.; Pulleyblank, B.; Raghavan, P.; and Sudan, M. 1994. The minimum latency problem. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, 163–171.

Croes, G. 1958. A method for solving traveling-salesman problems. *Operations Research* 6:791–812.

			Initial Allocation using SSI			Initial Allocation using SSC $\|C\| = \frac{1}{2}\|T\|$			Initial Allocation using SSC $\|C\| = \frac{2}{3}\|T\|$		
Capacity	Robots	Tasks	Initial Cost	Final Cost $K = \frac{1}{2}\|T_r\|$	Final Cost $K = \frac{2}{3}\|T_r\|$	Initial Cost	Final Cost $K = \frac{1}{2}\|T_r\|$	Final Cost $K = \frac{2}{3}\|T_r\|$	Initial Cost	Final Cost $K = \frac{1}{2}\|T_r\|$	Final Cost $K = \frac{2}{3}\|T_r\|$
4	4	16	2197	2225 (-1.3%)	2194 (0.1%)	2356	2286 (3.0%)	2307 (2.1%)	2367	2276 (3.9%)	2272 (4.0%)
4	6	24	2648	2642 (0.2%)	2684 (-1.4%)	2905	2736 (5.8%)	2801 (3.6%)	2696	2696 (0.0%)	2780 (-3.1%)
4	8	32	3070	3017 (1.7%)	3003 (2.2%)	3245	3131 (3.5%)	3125 (3.7%)	3233	3173 (1.8%)	3078 (4.8%)
4	10	40	3289	3133 (4.8%)	3190 (3.0%)	3599	3309 (8.1%)	3296 (8.4%)	3266	3200 (2.0%)	3238 (0.9%)
5	4	20	2578	2551 (1.0%)	2529 (1.9%)	2729	2685 (1.6%)	2657 (2.6%)	2670	2652 (0.7%)	2603 (2.5%)
5	6	30	3027	2968 (2.0%)	3033 (-0.2%)	3293	3192 (3.1%)	3175 (3.6%)	3189	3063 (4.0%)	3021 (5.3%)
5	8	40	3491	3392 (2.8%)	3407 (2.4%)	3702	3467 (6.4%)	3478 (6.1%)	3467	3447 (0.6%)	3543 (-2.2%)
5	10	50	3627	3574 (1.5%)	3567 (1.7%)	3859	3671 (4.9%)	3669 (4.9%)	3745	3591 (4.1%)	3635 (2.9%)
6	4	24	2795	2806 (-0.4%)	2765 (1.1%)	3101	2939 (5.3%)	2891 (6.8%)	2893	2858 (1.2%)	2846 (1.6%)
6	6	36	3353	3259 (2.8%)	3375 (-0.7%)	3599	3386 (5.9%)	3474 (3.5%)	3457	3406 (1.5%)	3371 (2.5%)
6	8	48	3748	3716 (0.8%)	3659 (2.4%)	4092	3740 (8.6%)	3810 (6.9%)	3975	3767 (5.2%)	3779 (4.9%)
6	10	60	4051	3973 (1.9%)	4006 (1.1%)	4315	4065 (5.8%)	4074 (5.6%)	4321	4121 (4.6%)	4160 (3.7%)

Table 1: Mean MiniSum Team Objective Results (percentage improvement of final cost compared to initial cost in brackets).

			Initial Allocation using SSI			Initial Allocation using SSC $\|C\| = \frac{1}{2}\|T\|$			Initial Allocation using SSC $\|C\| = \frac{2}{3}\|T\|$		
Capacity	Robots	Tasks	Initial Cost	Final Cost $K = \frac{1}{2}\|T_r\|$	Final Cost $K = \frac{2}{3}\|T_r\|$	Initial Cost	Final Cost $K = \frac{1}{2}\|T_r\|$	Final Cost $K = \frac{2}{3}\|T_r\|$	Initial Cost	Final Cost $K = \frac{1}{2}\|T_r\|$	Final Cost $K = \frac{2}{3}\|T_r\|$
4	4	16	826	821 (0.6%)	831 (-0.6%)	1014	935 (7.8%)	876 (13.6%)	913	908 (0.6%)	895 (2.0%)
4	6	24	750	761 (-1.6%)	775 (-3.4%)	993	854 (14.0%)	820 (17.4%)	822	812 (1.2%)	756 (8.0%)
4	8	32	711	716 (-0.8%)	707 (0.5%)	892	774 (13.2%)	720 (19.2%)	829	774 (6.7%)	768 (7.4%)
4	10	40	618	625 (-1.1%)	632 (-2.3%)	819	742 (9.4%)	679 (17.1%)	728	661 (9.2%)	682 (6.3%)
5	4	20	919	940 (-2.4%)	933 (-1.6%)	1097	967 (11.9%)	972 (11.4%)	999	966 (3.4%)	982 (1.7%)
5	6	30	841	840 (0.0%)	862 (-2.6%)	1012	902 (10.9%)	875 (13.5%)	930	857 (7.8%)	839 (9.7%)
5	8	40	773	776 (-0.5%)	760 (1.6%)	901	781 (13.2%)	773 (14.1%)	821	768 (6.4%)	747 (9.0%)
5	10	50	665	694 (-4.4%)	651 (2.0%)	815	746 (8.4%)	711 (12.8%)	764	726 (5.0%)	705 (7.6%)
6	4	24	1026	1049 (-2.2%)	1022 (0.4%)	1213	1116 (8.0%)	1028 (15.2%)	1028	1032 (-0.3%)	985 (4.2%)
6	6	36	886	919 (-3.8%)	915 (-3.4%)	1122	950 (15.4%)	950 (15.4%)	954	931 (2.5%)	854 (10.5%)
6	8	48	779	803 (-3.1%)	785 (-0.8%)	994	848 (14.7%)	827 (16.8%)	890	813 (8.7%)	811 (8.9%)
6	10	60	707	755 (-6.8%)	745 (-5.4%)	888	764 (14.0%)	721 (18.8%)	863	760 (11.9%)	734 (15.0%)

Table 2: Mean MiniMax Team Objective Results (percentage improvement of final cost compared to initial cost in brackets).

Dias, M., and Stentz, A. 2000. A free market architecture for distributed control of a multirobot system. *Proc. of the Int'l Conf. on Intelligent Autonomous Systems* 115–122.

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.

Elango, M.; Nachiappan, S.; and Tiwari, M. K. 2011. Balancing task allocation in multi-robot systems using k-means clustering and auction based mechanisms. *Expert Systems with Applications* 38(6):6486 – 6491.

Heap, B., and Pagnucco, M. 2011. Sequential single-cluster auctions for robot task allocation. *AI 2011* LNAI 7106:412–421.

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. *Proc. AAAI-06*.

Koenig, S.; Tovey, C.; Zheng, X.; and Sungur, I. 2007. Sequential bundle-bid single-sale auction algorithms for decentralized control. *Proc. IJCAI-07* 1359–1365.

Koenig, S.; Keskinocak, P.; and Tovey, C. 2010. Progress on agent coordination with cooperative auctions. In *Proc. AAAI-10*.

Lagoudakis, M.; Markakis, E.; Kempe, D.; Keskinocak, P.; Kleywegt, A.; Koenig, S.; Tovey, C.; Meyerson, A.; and Jain, S. 2005. Auction-based multi-robot routing. *Proc. Int. Conf. on Robotics: Science and Systems* 343–350.

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

Puig, D.; Garcia, M.; and Wu, L. 2011. A new global optimization strategy for coordinated multi-robot exploration: Development and comparative evaluation. *Robotics and Autonomous Systems*.

Solanas, A., and Garcia, M. 2004. Coordinated multi-robot exploration through unsupervised clustering of unknown space. In *Proc. IROS-04*, 717–721.

Tovey, C.; Lagoudakis, M.; Jain, S.; and Koenig, S. 2005. The generation of bidding rules for auction-based robot coordination. *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III* 3–14.

Zheng, X., and Koenig, S. 2009. K-swaps: Cooperative negotiation for solving task-allocation problems. In *Proc. IJCAI-09*, 373–379.