

# Market-based Coordination Strategies for Large-scale Multi-Agent Systems \*

MyungJoo Ham      Gul Agha

Open System Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign  
201 N. Goodwin Ave., Urbana, IL 61801, USA  
Email: {ham1, agha}@cs.uiuc.edu

**Abstract:** This paper studies market-based mechanisms for *dynamic coordinated task assignment* in large scale agent systems carrying out *search and rescue missions*. Specifically, the effect of different auction mechanisms and swapping are studied. The paper describes results from a large number of simulations of *homogeneous* agents, where by homogeneous we mean that agents in a given simulation use the same strategy. The information available to agents and their bidding strategies are used as simulation parameters. The simulations provide insight about the interaction between the strategy used by individual agents and the market mechanism. Performance is evaluated using several metrics: mission time, distance traveled, communication and computation costs, and workload distribution. Some of the results obtained include: limiting information may improve performance, different utility functions may affect the performance in non-uniform ways, and swapping may help improve the efficiency of assignments in dynamic environments.

**Keywords:** auction, market-based approach, multi-agent system, task assignment.

## 1. Introduction

New types of physical agents being developed include robots, small unmanned aerial vehicles (micro-UAVs), and unmanned underwater vehicles (UUVs). Such physical agents will be useful for surveillance, search and rescue, mine sweeping and other applications. This paper focuses on a two dimensional search and rescue (SR) problem involving pursuer robots and mobile targets. In SR, the number of tasks will generally exceed that of agents and a task may require multiple agents. Thus, efficient methods, which can enable coordination between the agents, are required. Note that the SR problem is computationally intractable. Even a simplified version of the SR problem, the vehicle routing problem [7], is NP-hard.

To address the SR problem, two mechanisms are investigated: *auctions* and *swapping*—both fully *distributed* and *asynchronous*. These methods have reasonable computational complexity. Each target requires a number of auctions to find robots to serve. In practice, this is a small number. If the communication and sensing ranges are bounded, the mechanisms require constant time for each round of auctions. If the ranges are unbounded,  $O(n)$  time is required, where  $n$  is the number of tar-

get and robot agents combined. However, note that the auctions that are used in this paper yield sub-optimal assignments [3].

Previous research [1][4] and other similar work [2][5][6][8][9][10][12][13] studied the problem with small-scale experiments. However, small-scale experiments can be easily biased by specific experimental parameters. Another limitation is that they do not establish the scalability of mechanisms.

These limitations motivate us to run large-scale simulations in which the strategies and experimental parameters are varied. We study different auction mechanisms, non-cooperative heuristic method, which resembles *swarm intelligence* [11], as a control and swapping. Different bidding strategies, which weigh the utility, cost, and popularity of a target, are used. Experimental parameters are varied in order to test the robustness and characteristics of the mechanisms.

Limiting sensing and communication ranges can provide more scalability in real applications because of the broadcasting cost. Perhaps more surprisingly, the results suggest that limiting the ranges improves performance in most of the metrics.

In theory, assuming a fixed order of synchronized auctions, all auction mechanisms would yield the same results. However, in this paper, the assumption is not met and it is easy to see how different auction mechanisms may produce different results.

Two robots may swap tasks to reduce their costs. Dynamism of the environment and asynchrony of auctions create the need for swapping although swapping causes additional delays and messages.

The rest of this paper is organized as follows. Section 2 describes previous research on multi-agent coordinations. Section 3 describes the methods studied. Section 4 describes the results. Section 5 analyzes these results. Finally, Section 6 discusses the conclusions and directions for future research.

## 2. Related Work

In principle, a centralized approach can provide results that are equal to or better than those of distributed approaches [3][10]. However, it has a single point of failure and requires fully connected networks. More critically, it is not scalable.

A number of distributed approaches have also been proposed. Most of the proposed mechanisms are market-based. Some of the mechanisms are offline algorithms, i.e., the assignment is done once at the beginning [13]. Obviously, offline algorithms cannot adapt to dynamic environments. Other research has studied online mechanisms [1][2][6][8][9]. Swarm

\* This research has been supported in part by NSF under grant CNS 05-09321 and by ONR under DoD MURI award N0014-02-1-0715. We thank Liping Chen and Rajesh Kumar at UIUC, Tom Brown at Google, and the anonymous reviewers of the paper for their invaluable comments and advice.

intelligence [11], analogous to the behavior of social insects, is an example of non-market based mechanism.

Different degrees of dynamicity have been experimented; tasks may be static, passive or dynamic. *Static tasks* do not change their utility or cost: [2][3][6][13]. *Passive tasks* may be modified only by the action of robots: [5][8][9]. *Dynamic tasks* can be modified by themselves: [1][4]. *Preemption* to change an agent's attention and *adaptation* become important in order to let agents respond to the change of dynamic environment.

The previous research [1] proposed forward/reverse auction and swapping for task allocation with physical agents in dynamic environment. However, the main weakness of [1] is that the experiments were not sufficient—only one execution was carried out for each method in a small-scale experiment with a single metric (mission time). Moreover, the effect of the bidding strategy was not examined.

Distributed multi-robot coordination research based on a geographic area usually uses small-scale experiments. The problem size is extended to show the scalability, test various coordination methods, and experiment more concretely with more metrics and various experimental parameters. The behavior of agents, such as bidding, cost evaluation, and the dimensions and mobility of agents are configured by the values similar to those in [4], which are based on the physical simulations in [1].

### 3. Method

A number of simplifying assumptions are made. Agents are on a bounded rectangular Euclidean plane. Agents in a given simulation are homogeneous. Robots observe every target within the sensing range and notify others in the communication range about its own observed targets. Targets move around with predefined patterns, which are not predicted by robots. Although other algorithms such as roaming algorithms and movement prediction may improve the performance, they are not studied, as their benefits are likely to be marginal and the purpose is to focus on the effect of coordination mechanisms.

In order to be served, a target  $t$  requires multiple dedicated  $req_t$  ( $>1$ ) robots to be present nearby ( $<0.2m$ ) at the same time.  $t$  distributes its utility  $util_t$  evenly  $util_t/req_t$  to each of the  $req_t$  robots; the  $req_t+1$ 'th cannot receive the payoff.

Each instance of the problem is defined as a *mission*; a mission is complete when every target agent has been served. The number of targets is large enough for each robot to serve multiple times. Thus, a robot may participate in several auctions.

#### 3.1. Coordination Methods

We study several methods for coordination between agents, including non-interactive methods, auctions, and swapping.

**Non-cooperative Heuristic Method (N/C).** A robot agent  $r$  chooses the target agent  $t$  that has the largest expected profit. The expected profit is  $util_t/req_t - cost_{NC}(r, t)$ , where  $cost_{NC}(r, t)$  is the N/C's cost for  $r$  to serve  $t$ , which is the distance and pivoting cost. A robot changes its target if another target becomes more attractive.

**Forward Auction.** A robot agent  $r$  bids for the target agent that has the largest expected profit,  $f_{util}(t) - cost(r, t) - price_t$ , where  $f_{util}(t)$  is a utility function described later. A bid is retracted if another target becomes more attractive. An auction is managed by an auctioneer, which is one

of the bidders. An auction of  $t$  is finished if it has enough number of bidders ( $\geq req_t$ ) after  $T_{round}$ .

When an auction for a target agent  $t$  is started, the auctioneer accepts bids higher than  $price_t$  of Eq. (1).

$$price_t = \begin{cases} \min(\min\_bid, max\_rej\_bid), & asn_t > 0; \\ max\_rej\_bid, & otherwise. \end{cases} \quad (1)$$

$asn_t$  is the number of bidders,  $\min\_bid$  is the lowest bid, and  $max\_rej\_bid$  is the larger value of the highest rejected bids and the initial price. If  $asn_t > req_t$ , the lowest bid is rejected until  $asn_t = req_t$ . The rejected bidder bids for  $t$  again if it can outbid with the updated conditions. Otherwise, it searches for another target.

An auction is stopped after  $T_{round}$ , a specified time period from the beginning of the auction. If the auction does not result in a sufficient number of bidders, the bidders are released and the auction is paused and restarts after a random interval ( $random(0.1, 1.0) \cdot T_{round}$ ) in the experiments. After the pause, the auction restarts. This delay allows the environment to evolve (e.g. more robots to become free).

**Reverse Auction.** In contrast to a forward auction, where robots increase prices to attract targets, in a reverse auction, targets decrease prices to attract robots. The price is cut when the auction is paused. The price is also cut if a bidder retracts when  $asn_t \leq req_t$ . Unlike a forward auction, higher bids do not raise prices.

**Forward/Reverse Auction (F/R).** Using both forward and reverse auction in order to reduce auction delay with equivalent auction results has been proposed by [1][3]. Forward/reverse auction is implemented by running forward auction during normal operations and reverse auction when the auction is paused or a bid is retracted.

**Sealed-bid Auction based on F/R Auction.** Unlike auctions with actual fund transactions, robots do not actually pay anything to win auctions. Thus, target price may not be a cost factor. Given this, a sealed-bid auction is designed based on a forward/reverse auction. The expected profit is  $f_{util}(t) - cost(r, t)$ . However, bids are rejected based on the price as other auction methods do.

#### 3.2. Cost Function

Cost function  $cost(r, t)$  includes the distance between robot agent  $r$  and target agent  $t$ , bidding cost, bid retraction cost, redundant auction cost, and travel related costs such as pivoting and collision avoidance [4]. Of these factors, generally the distance between robot and target is the decisive factor.

#### 3.3. Utility Functions

We describe various utility functions that a robot agent uses in determining how much to bid. Essentially, these functions determine the bidding strategy of a robot.

**Default (Static).** The *Default* utility function of a target agent  $t$  is Eq. (2), which is the payoff that each robot receives after serving  $t$ . It is also called *Static* utility function because the value never changes.

$$f_{util}(t) = util_t/req_t \quad (2)$$

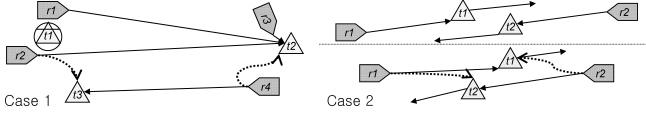


Fig. 1. Examples when swapping is beneficial.

**Division.** Although the default utility function reflects the exact value of serving a target, it may not be sufficient because the status of an auction is not included. For example, let's assume that there are targets  $t_1$  and  $t_2$  with  $req_{t_1} = req_{t_2} = 5$ ,  $asn_{t_1} = 4$ ,  $asn_{t_2} = 1$ , and robot  $r$  needs to choose either  $t_1$  or  $t_2$ . Then,  $r$  may want to prioritize  $t_1$  because  $t_1$  needs only one more bidder while  $t_2$  needs four more. The *Division* utility function Eq. (3) is designed to increase  $f_{util}(t)$  as  $asn_t$  increases and to increase more if  $asn_t$  is near  $req_t$ . This was used in the previous work [1].

$$f_{util}(t) = \frac{util_t}{req_t - \min(asn_t, req_t - 1)} \quad (3)$$

**Division-Restricted.** The division utility function results in a hoarding problem. If target  $t$  already has enough bidders ( $asn_t \geq req_t$ ),  $f_{util}(t)$  is still boosted and  $t$  can attract more bidders. This over attraction can increase auction cost because a target can attract too many bidders; neighbor target agents suffer from starvation. To mitigate this problem, the  $f_{util}(t)$  uses the default utility function when the bidder is not yet assigned to  $t$  and  $asn_t \geq req_t$ .

**Division-Small.** When targets  $t_1$  and  $t_2$  ( $req_{t_1} = 5$ ,  $asn_{t_1} = 4$ ,  $req_{t_2} = 2$ ,  $asn_{t_2} = 1$ ,  $\forall t: util_t/req_t = 10$ ) are close to a robot  $r$ , the utility values of  $t_1$  and  $t_2$  should be the same because both  $t_1$  and  $t_2$  need only one more robot and the two targets provide the same utility for each robot ( $util/req$ ). However, division and division-restricted utility functions give the values differently;  $f_{util}(t_1) = 50$  and  $f_{util}(t_2) = 20$ . With *Division-Small* utility function, maximum possible boost ratio is the same regardless of  $req_t$ ;

$$f_{util}(t) = \frac{util_t}{req_t} \cdot \left(1 + \frac{1}{req_t - \min(asn_t, req_t - 1)}\right) \quad (4)$$

**Division-Restricted and Small.** This is a combination of division-restricted and division-small.

**Linear.** In division methods, the utility value for target  $t$  is boosted more when  $asn_t$  is closer to  $req_t$ . However, with *Linear* utility function Eq. (5), the utility boost per bidder is constant.

$$f_{util}(t) = util_t/req_t \cdot (1 + \min(asn_t, req_t)/req_t) \quad (5)$$

### 3.4. Swapping

Assignments may become obsolete because there are a series of asynchronous auctions and targets are moving.

Case 1 of Fig. 1 is an example when the asynchrony of auctions makes swapping attractive ( $req_{t_1} = 2$ ,  $req_{t_2} = 3$ ,  $req_{t_3} = 1$ ). After serving  $t_1$ ,  $r_1$  and  $r_2$  are assigned to  $t_2$  along with  $r_3$ .  $r_4$  is assigned to  $t_3$  before  $t_1$  is served.  $r_4$  could not be assigned to  $t_2$  because  $r_1$  and  $r_2$  were serving  $t_1$ . However, it is obvious that if  $r_2$  and  $r_4$  swap their tasks, the costs can be reduced.

Case 2 is an example where the dynamicity causes the need for swapping. Here  $r_1$  was pursuing  $t_1$  and  $r_2$  was pursuing  $t_2$ . However, as  $t_1$  and  $t_2$  move, the best targets of  $r_1$  and  $r_2$  change. If  $r_1$  and  $r_2$  swap tasks, their costs can be reduced.

Table 1. Simulation data set.

Name	# Robot	# Target	Field Size	Sensing/Comm Ranges
Dense	250	750	40x40 ( $m$ )	300/1200
Corner	50	500	45x45 ( $m$ )	300/1200
Scatter	50	500	45x45 ( $m$ )	300/1200
G.Corner	50	500	45x45 ( $m$ )	Global
G.Scatter	50	500	45x45 ( $m$ )	Global

The robot agent that requests a swap becomes a swapper, which chooses a swapee. A swap should be beneficial for both swapper and swapee. A swapper  $r_1$  with target  $t_1$  sends a swap request to swapee  $r_s$  with target  $t_s$  when the expected benefit  $EB(r_1, r_s) > 0$  and  $\forall i: EB(r_1, r_s) \geq EB(r_1, r_i)$ , where

$$EB(r_1, r_i) = cost_s(r_1, t_1) + cost_s(r_i, t_i) - cost_s(r_1, t_i) - cost_s(r_i, t_1) - TH_{swap} \quad (6)$$

,  $t_i$  is  $r_i$ 's target, and  $TH_{swap}$  is the swap threshold.  $cost_s(r, t)$  is the cost function for the swap decision, which corresponds to the distance and azimuth difference between  $r$  and  $t$ .

## 4. Experiments

The simulation is implemented in C++ using Repast.NET [14] and executed on Intel Core 2 systems with Windows XP. As mentioned earlier, agents are programmed to behave like the physical agents in [1].  $util_t/req_t$  is defined by the field size so that it is always larger than distance costs and every target has the same utility per robot.  $\forall t: 2 \leq req_t \leq 5$ . Mission time (time spent to complete a mission), movement distance (total distance covered by robots), auction delay, number of messages, and fairness (load balance) are measured.

Several data sets (Table 1) are experimented. The data set *Dense* represents a field where each target agent almost always has enough number of robot agents nearby so that virtually no coordination between robot agents is required. In *Corner* and *G.Corner*, robot agents start from a corner area and the density of robot agents is lower. In *Scatter* and *G.Scatter*, robot agents are scattered in the field. In the lower-density cases, servicing targets is difficult without coordination.

A mission is stopped when 90% of targets are served. Otherwise, the results would be distorted by the last few targets; it often takes far greater time to find last few targets if the ranges are limited. Each mission is run 15 times for every combination of 5 data sets, 5 coordination methods, and 6 utility functions. Fig. 2, 3, and 4 show performance improvement over controls. Positive values mean improvement and negative values mean deterioration except “-Messages”, where it is negated. Confidence intervals use a 95% level.

N/C and forward auction with dynamic utility functions suffer from deadlock. Reverse auction shows no merit over forward auction. Some of the dynamic utility functions with forward/reverse auction do not complete missions before the simulation time limit.

In *Dense*, N/C finishes missions and performs well. Sealed-bid auction with default utility function and swapping, which performed best in *Dense*, performs 7% better in mission time and 11% better in movement distance than N/C. Some cases of Forward or F/R auctions even perform worse than N/C in *Dense*. However, N/C does not perform well in the low density data sets. In *Corner*, N/C serves 53% of targets before the time limit; other methods finish in about half of the time limit. N/C

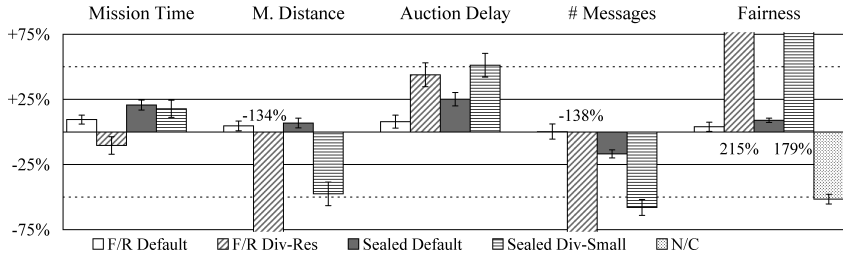


Fig. 2. Improvement over forward auction and default utility function.

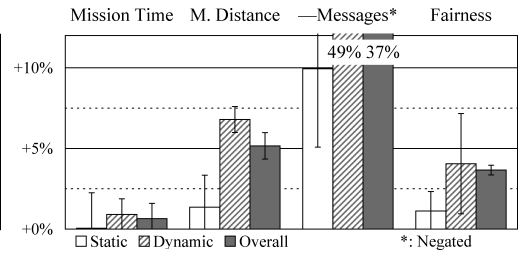


Fig. 3. Performance improvement by swapping.

suffers from deadlock in Scatter and G.Scatter and is much slower than other methods in G.Corner.

Fig. 2 shows performance improvement, which are shown by values compared to those of forward auction and default utility function. F/R auction improves performance in all metrics. Sealed-bid auction improves performance further except for the number of messages. N/C is shown with a fairness metric only because it cannot complete missions in most cases.

Dynamic utility functions with forward auction suffer from deadlock, which was not observed in the previous work with small-scale experiments [1]. Although no deadlock is found, dynamic utility functions do not perform well with F/R auction; ping-pong bidding, bidder alternatively bids and retracts (see Section 5 below) is observed. On the other hand, dynamic utility functions successfully reduce auction delay with sealed-bid auction except when the division utility function is used. However, the reduction in auction delay is often not enough to reduce mission time because dynamic utility functions damage auction quality, which in turn increases movement distance; dynamic utility functions provide trade-offs between auction delay and auction quality. Fairness is improved significantly with dynamic utility functions.

Fig. 3 shows the performance improvement by swapping. Auction delay is not shown because swapping does not affect the auction process. The number of messages is increased (deteriorated) by swapping significantly. The number of messages is increases more with dynamic utility functions ( $10\% < 49\%$ ), which incur more swapping ( $478 > 17$ ). Fairness is improved with swapping, which implies that swapping helps load balancing. In Dense, the number of messages increases up to 77%, which implies that the communication overhead of swapping can be almost as much as that of an auction. The high communication overhead of swapping makes it less efficient as it can increase not only the number of messages, but also mission time if the communication delay is significant. The negative effect of swapping on mission time can be inferred by the result in Fig. 3; even though the movement distance is reduced, mission time is not as reduced. Performance improvement is more significant with dynamic utility functions, which have poor auction quality compared to a static utility function. Although the auction quality of dynamic utility function improves with swapping, swapping does not make movement distances (auction quality) as short as those obtained by using a static utility function.

Fig. 4 shows how limiting sensing and communication ranges affects the performance by comparing the result of limited ranges with that of global ranges. Using a static utility function, limiting ranges reduces auction delay by 28% and mission time by 4%. However, movement distance and the number of messages increase with limited ranges. Because each auction has more information with global knowledge,

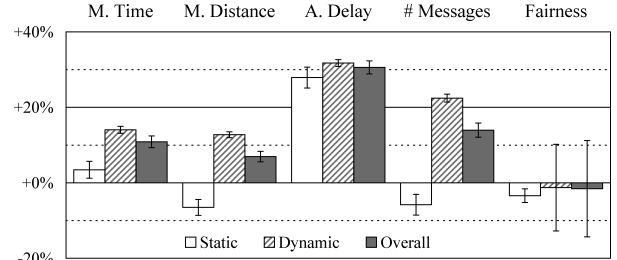


Fig. 4. Improvement by limiting sensing/comm ranges.

having longer distance with limited ranges seems to be natural. However, having too much information may lead to more delay and damages the overall performance as Fig. 4 suggests. Using dynamic utility functions, limiting ranges enhances the performance in four metrics significantly. Limiting the ranges can be interpreted as filtering objects because greedy algorithms such as auction methods, usually try to choose targets near the robots; thus, the probability to coordinate with robots far away is relatively low. Fairness is reduced little with static utility and does not have significant differences in overall.

The improvement achieved by various methods using limited ranges is compared with the improvement using global ranges. The results show that using limited ranges, movement distance and number of messages are reduced more except for F/R auction with static utility function compared with forward auction with static utility function, which suggests that it is easier to improve performance (with limited ranges). Using dynamic utility functions, the performance improvement difference from the cases of global ranges is more significant.

Both the number of swaps and the sum of estimated swapping benefit ( $\sum EB(r_1, r_i)$ ) are larger with limited ranges using static utility function. However, using dynamic utility functions and limiting ranges makes the two values smaller. More swapping implies that the auction results have poor quality. Thus, we can infer that limited ranges (as opposed to global ranges) gives less efficient plans with static utility function and gives more efficient plans with dynamic utility functions; movement distance in Fig. 4 shows the same tendency.

## 5. Discussion

The non-cooperative heuristic method (N/C) often suffers from deadlock unless the density of robot agents is very high (as in Dense) or the  $req_t$  values are small enough. Fig. 5 shows an example. Because each robot agent may find its own best target differently, this deadlock is not unlocked unless additional robot agents come in or the targets move so that the robots may change their targets. However, unless the density of robot agents is sufficiently high, unlocking does not happen frequently enough to accomplish missions.

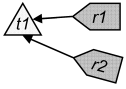


Fig. 5. N/C suffering from deadlock.  
 $req_{t1} = req_{t2} = 3$ .

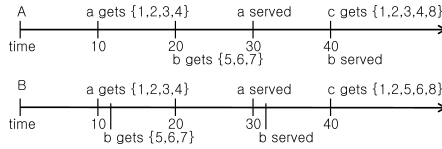


Fig. 6. Inequivalent of auction sequences with different convergence speed.

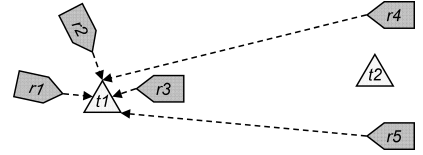


Fig. 7. Distorted utility value with dynamic utility function.

Table 2. Starvation with forward auction.

Time	$f_{util}(t1)$	$price_{t1}$	Actions
0	1000	100	$r1$ bids “100”.
1	1500	100	$r2$ bids “1200”.
2	2000	1200	$r2$ retracts.
3	1500	1200	$r1$ retracts.
4	1000	1200	$t1$ 's price is too high.

Forward auction with dynamic utility functions also suffers from deadlock, which is not found in the small-scale experiments of the previous work [1]. When a target price increases according to the positions of agents or auction subscription status, the target can suffer from starvation, which in turn creates deadlock if many targets starve. This happens frequently with dynamic utility functions, as such functions can cause greater increases in target prices. Table 2 shows an example. Such starvation may happen without dynamic utility function; it is observed in preliminary simulations when targets moved away too fast and robots could not finish an auction in time. Adding a reverse auction to a forward auction helps in adapting to the dynamic environment, which prevents the starvation, as well as in reducing convergence time.

Adding a reverse auction to a forward auction can alter the assignments when targets are dynamic or robots have a series of asynchronous auctions—each auction is done asynchronously with a different duration. The auctions are supposed to have the same results with different delay [3]: forward, reverse, and F/R auction. However, if targets move, their positions, which determines the costs, are different according to each auction method, which in turn makes the assignments different.

Even if targets are static, assignments from a second round can be different because each robot has multiple asynchronous auctions. Fig. 6 shows an example with targets  $\{a, b, c\}$ ,  $req_a=4$ ,  $req_b=3$ ,  $req_c=5$ , and robots 1 to 8 using different auction methods (i.e. F/R versus forward); the two auction results are different. Because the system is asynchronous and distributed, each robot incurs additional costs to wait for other auctions to converge and other robots to complete their services. In a fully distributed system with a dynamic environment, such costs can be too high. The properties of a target changes while robots are sending “confirmation of convergence”, subsequently, the robots may need to cancel their confirmations and restart auctions because of the changes in the environment. As a result, given a dynamicity and asynchrony, distributed auctions may need to wait indefinitely in order to find an equilibrium; thus, auctions cannot wait for an equilibrium. Nonetheless, adding a reverse auction to a forward auction has the advantage of converging with less delay and fewer messages. Although adding a reverse auction may give different results, it does not necessarily mean worse results; the results show that the auction quality is slightly improved. This

Table 3. Ping-pong bidding with F/R auction.

Time	$EP_{t1}$	$price_{t1}$	$EP_{t2}$	$price_{t2}$	Actions
0	1010	100	1000	100	$r$ bids $t1$
1	100	1010	700*	100	$r$ retracts bid
2	210	800	1000	100	$r$ bids $t2$
3	-90*	800	100	1000	$t1$ discounts
4	310*	400	100	1000	$r$ retracts bid
5	610	400	300	800	$r$ bids $t1$
6	100	1010	0*	800	$t2$ discounts

\*: cost of retracting a bid applied.  $EP$ : expected profit.

may be a result of the fact that a faster auction can respond faster in a dynamic environment.

A F/R auction can suffer from ping-pong bidding; a bidder alternatively bids to multiple targets and auctions are delayed. The target price in a F/R auction can go up and down. If the target price fluctuates, bidders may alternatively bid to different targets without completing an auction. Table 3 shows an example. Ping-pong bidding usually stops either by another bidder or by a completion of the auctions. Applying an additional cost for retracting a bid mitigates the problem a little. However, as shown on Table 3, ping-pong bidding still happens with the cost of retracting a bid. If the cost of retracting a bid is too high the algorithm cannot adapt to the change in the environment. Ping-pong bidding happens more frequently if dynamic utility functions are used because they cause greater fluctuations in the target price. With sealed-bid auctions, ping-pong bidding does not occur because bidders do not see the target price.

Sealed-bid auctions perform best in terms of most of the performance metrics. Sealed-bid auctions may have a disadvantage because they do not consider the target price as a cost factor, which reflects how others bid; the more popular a target is, usually the more expensive it is. However, because robot agent in the system does not actually pay the target price, but incurs costs such as movement distance and pivoting, price may be neglected by robots. The results show the best performance with sealed-bid auction in mission time, movement distance, auction delay, and fairness.

Dynamic utility functions can distort a target’s utility, which can make assignments inefficient by exaggerating the value of a target that has a higher cost. This disadvantage makes movement distance longer than default(static) utility function. Fig. 7 shows an example of distorted utility, where  $req_{t1} = 4$ ,  $req_{t2} = 1$ , and  $\forall i : util_{ti}/req_{ti} = 10$ . When the utility of  $t1$  is boosted enough to ignore the distances by  $r1$ ,  $r2$ , and  $r3$ , both  $r4$  and  $r5$  bid for  $t1-t2$  is better. Although dynamic utility functions result in longer movement distance, shorter auction delay may compensate for the increased movement.

A pair of robot agents can repeatedly swap with each other: *ping-pong swapping*. Swap threshold  $TH_{swap}$  reduces the frequency of ping-pong swapping. However  $TH_{swap}$  cannot completely eliminate ping-pong swapping and  $TH_{swap}$  also reduces

Table 4. Summary of strategies.

Strategy	Results
Coordination method	
N/C	Deadlock with non-trivial problems
Forward	Deadlock with dynamic utility functions
Reverse	No merit over Forward
F/R	Better than Forward. Ping-pong bidding.
Sealed-bid	Best except for the number of messages.
Utility function	
Static	Shorter movement distance
Dynamic	Shorter auction delay
Tradeoff between convergence speed and solution quality	
Swapping	
Performance improved. Ping-pong swapping occurs.	

the frequency of beneficial swaps. Ping-pong swapping is caused either by collision avoidance or by the asynchrony and delay of agents. When ping-pong swapping happens, both mission time and movement distance increase. Even when ping-pong swapping does not happen because of the threshold, the initial swap may have already made the routing plans less efficient. Because some swaps are useless or harmful, the performance improvement of swapping is not as significant as one would expect.

## 6. Future Research

Various auction mechanisms and swapping for distributed multi-robot coordination, and with robots employing different bidding strategies in large-scale multi-agent systems are experimented with. The results suggest that the mechanisms work with dynamic environment in large-scale systems and the mechanisms can work with larger-scale systems because the mechanisms perform well with limited sensing and communication ranges. Table 4 summarizes the results. However, bidding strategies, which are expressed by utility functions, may include more factors such as the number of potential bidders and the number of available targets nearby (potential utility). Besides, there is a possibility that it may perform better if robot agents are heterogenous and have mixed strategies.

Various simulation parameters are also experimented with: sensing and communication ranges, robot agent density, and initial positions of robot agents. Table 5 summarizes the results. However, we may need to experiment further by varying more parameters such as different field sizes, static targets, and varying target speeds. This may help verify the characteristics and adaptability of the mechanisms.

Additional metrics such as the number and benefit of effective swaps, number of ping-pong bids, robot idle time, and statistics of price-cut may help measure the performance more concretely; thus, we can find how to improve the mechanisms and verify the conjectures to explain symptoms such as ping-pong bidding and ping-pong swapping.

Forward, reverse, and forward/reverse auctions are known to be equivalent under certain restrictive assumptions [3]. These assumptions are that the auctions for all tasks (targets) are simultaneous, the tasks are static, and that each agent bids in only one auction (and for a predetermined task). In this case, the auction is used as an offline tool.

In general, the auction methods do not yield the same results (equilibria) under the looser conditions of this paper: the environment is dynamic and a sequence of asynchronous auctions is used. However, it is possible that the auction methods may

Table 5. Summary of simulation parameters.

Parameters	Results
Sensing/communication ranges	
Limited	Auction methods perform better.
Global	N/C suffers from deadlock less severely.
Robot agent density	
Dense	Both N/C and auction methods work.
Sparse	N/C fails. Auction methods perform well.
Initial positions of robot agents	
Corner	Less performance gap between strategies.
Scattered	N/C suffers from deadlock more severely.

result in the same sets of possible assignments even though they result in different assignments. Proving that the set of possible assignments under different auction methods is the same with asynchronous auctions remains an open problem. However, even if the set of possible solutions is equal, the problem of determining the speed of convergence and the likelihood of better solutions is a more difficult problem. Given the differences that are apparent in the simulations, we conjecture that different auction mechanisms are not equivalent. This question would be much harder to resolve analytically.

## References

- [1] A Ahmed, A Patel, T Brown, M Ham, M W Jang and G Agha, Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism. Proc. of KIMAS 2005, pp. 311–317.
- [2] C A Bererton, G J Gordon and S Thrun, Auction mechanism design for multi-robot coordination, In *Advanced in Neural Information Processing Systems 16*, S Thrun, L Saul, and B Schölkopf, Eds. MIT Press, Cambridge, MA, 2004.
- [3] D P Bertsekas and D A Castanon, A forward/reverse auction algorithm for asymmetric assignment problems. *Computational Optimization and Applications*, No. 3, 1992, pp. 277–297.
- [4] T D Brown, Decentralized coordination with crash failures. Master’s thesis, Univ. of Illinois at Urbana-Champaign, 2005.
- [5] L Chaimowicz, M F M Campos and V Kumar, Dynamic role assignment for cooperative robots. Proc. of ICRA 2002, Vol. 1, pp. 293–298.
- [6] P R Chandler, M Pachter, S R Rasmussen and C Schumacher, Multiple task assignment for a uav team. Proc. of AIAA GN&C 2002, AIAA-2002-4587.
- [7] G B Dantzig and J H Ramser, The truck dispatching problem. *Management Science*, Vol. 6, No. 1, 1959, pp. 80–91.
- [8] B P Gerkey and M J Mataric, Sold!: Auction methods for multirobot coordination. *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 5, 2002, pp. 758–768.
- [9] J Guerrero and G Oliver, Multi-robot task allocation strategies using auction-like mechanisms. *Artificial Intelligence Research and Development*, Vol. 100, 2003, pp. 111–122.
- [10] A Kothar, D C Parke and S Sur, Approximately-strategyproof and tractable multi-unit auctions. Proc. of ACM EC New York, NY, USA, 2003, ACM Press, pp. 166–175.
- [11] M J B Krieger, J B Billeter and L Keller, Ant-like task allocation and recruitment in cooperative robots. *Nature*, Vol. 406, 2000, pp. 992–995.
- [12] D Palmer, M Kirschenbaum, J Murton, K Zajac, M Kovacina and R Vaidyanathan, Decentralized cooperative auction for multiple agent task allocation using synchronized random number generators. Proc. of IROS 2003, Vol. 2, pp. 1963–1968.
- [13] W E Walsh and M P Wellman, A market protocol for decentralized task allocation. Proc. of ICMAS Washington, DC, USA, 1998, pp. 325–332.
- [14] Repast agent simulation toolkit, <http://repast.sourceforge.net>