2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

# Distributed Multi-Robot Task Assignment and Formation Control

Nathan Michael, Michael M. Zavlanos, Vijay Kumar, and George J. Pappas

*Abstract*— **Distributed task assignment for multiple agents raises fundamental and novel problems in control theory and robotics. A new challenge is the development of distributed algorithms that dynamically assign tasks to multiple agents, not relying on *a priori* assignment information. We address this challenge using market-based coordination protocols where the agents are able to bid for task assignment with the assumption that every agent has knowledge of the maximum number of agents that any given task can accommodate. We show that our approach always achieves the desired assignment of agents to tasks after exploring at most a polynomial number of assignments, dramatically reducing the combinatorial nature of discrete assignment problems. We verify our algorithm through both simulation and experimentation on a team of non-holonomic robots performing distributed formation stabilization and group splitting and merging.**

## I. Introduction

Recent advances in communication and computation have given rise to distributed control of multi-agent systems which, compared to conventional centralized control, provide increased efficiency, performance, and scalability as well as robustness due to its ability to adjust to agent failures or dynamically changing environments. Inspired by these appealing properties of distributed control, we propose a distributed and online solution to the multi-agent dynamic task allocation problem where the assignment of robots to tasks may need to be continuously adjusted depending on changes in the task environment or group performance.

Assignment problems are fundamental in combinatorial optimization and, roughly, consist of finding a minimum weight matching in a weighted bipartite graph. They arise frequently in operations research, computer vision, as well as distributed robotics, where graphs have recently emerged as a natural mathematical description for capturing interconnection topology [1]–[6]. Depending on the form of the cost function, assignment problems can be classified as linear or quadratic. Optimal solutions to the linear assignment problem can be computed in polynomial time using the Hungarian algorithm [7]. The quadratic assignment problem, however, is NP-hard [8] and suboptimal solutions are achieved by means of various relaxations. Approaches are either purely discrete [9], or continuous [10], based on the solution of differential equations that always converge to a discrete assignment.

This work is partially supported by ARO MURI SWARMS Grant W911NF-05-1-0219 and the NSF ITR Grant 0324977.

Nathan Michael and Vijay Kumar are with GRASP Laboratory, Department of Mechanical Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA {nmichael,kumar}@grasp.upenn.edu. Michael M. Zavlanos and George J. Pappas are with GRASP Laboratory, Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA {zavlanos,pappasg}@grasp.upenn.edu

In distributed robotics, task assignment considers an environment populated with tasks and a group of robots that are equally capable of performing each of the tasks but may only be assigned to one task at any given time. Tasks can in general be generic objects such as spatial locations, network resources, or classifications. Assignment approaches can be either online [11], [12] or off-line [13] depending on whether the assignment of agents to tasks is designed to account for changes in the task environment or is solved independently in advance. Task assignment also aims to enhance the system's performance, typically by reducing the overall execution time. In distributed multi-agent target assignment this usually translates to optimizing a cost associated with the total distance traveled by the agents [13], [14].

In this paper we propose a distributed market-based coordination algorithm in which agents are able to bid for task assignments with the assumption that the agents have knowledge of all tasks as well as the maximum number of agents that can be assigned to every individual task. Each auction is performed among neighboring groups of agents and requires only local communication. We assure that the final assignment will respect the maximum agent specification for every task by requiring that all agents bidding for a particular task are eventually neighbors. Although no global cost is optimized, we show that a desired assignment of agents to tasks is always achieved after exploring at most a polynomial number of assignments, dramatically reducing the combinatorial nature of discrete assignment problems. We illustrate the effectiveness of our approach through nontrivial computer simulations as well as through experimentation. In particular, we consider teams of multiple robots that navigate and stabilize to a given formation, or split and merge into groups of different sizes.

The presentation of the paper is organized as follows. In Sect. II we formalize the assignment problem and provide necessary notation. Sect. III details the distributed coordination algorithm. We address experimental and software details in in Sect. IV and algorithm performance in Sect. V. Simulation and experimental results for a team of differential drive non-holonomic robots are presented in Sect. VI.

## II. Problem Definition

Given $n$ identical agents and $m \leq n$ tasks to be accomplished by the agents, let $n_k$ be the maximum number of agents that can be assigned to task $k$, such that $n = n_1 + \cdots + n_m$. Then, the problem addressed in this paper can be stated as follows.

*Problem 1 (Distributed Task Assignment):* Given $n$ agents and $m$ tasks, determine distributed control laws that

split the agents into $m$ groups of size $n_k$ associated with each task $k = 1, \ldots, m$.

In the proposed framework, a *task* represents a generic notion associated with a common objective or item that all agents assigned to it should share. For instance, a task can be associated to a location that a group of agents should explore, a box that a group of agents should lift, or even a computation that a group of agents should collaboratively evaluate. In all cases, the common objective that defines the task requires collaboration of the group of agents that are assigned to it.

Since the assignment of the agents to tasks is not provided *a priori*, it needs to be determined *dynamically*. We achieve this goal by letting every agent explore a sequence of tasks it considers *available* and eventually establish an assignment with an available task, according to the specifications of Problem 1.[1] Task assignment is based on a *distributed market* characterized by the absence of a centralizing auctioneer, where agents are able to *bid* for the task to which they wish to be assigned. Local communication among neighbors enables assignments to be made by comparing bids as well as information propagation in the underlying network, regarding the *availability* of the requested tasks. Inter-agent communication and the proposed bidding process together form a *distributed coordination* framework able to achieve the desired assignment (Section III).

## III. Distributed Task Assignment

Let $\mathcal{I} = \{1, \ldots, m\}$ denote the index set of all available tasks and define the *availability* $\alpha_k(t) \in \{0, 1, \ldots, n_k\}$ of a task $k \in \mathcal{I}$ at time $t$ by the number of agents that can still be assigned to that task without exceeding the limit $n_k$. We assume that every agent $i$ holds an *estimate* of the availability $\alpha_k(t)$ for every task $k \in \mathcal{I}$, which we denote by $\alpha_k^{[i]}(t)$. For every agent $i$, let $\mathcal{I}_i^a(t) \triangleq \{k \in \mathcal{I} \mid \alpha_k^{[i]}(t) > 0\}$ denote the index set of all tasks that are considered *available* at time $t$. Then, $\mathcal{I}_i^t(t) \triangleq \mathcal{I} \backslash \mathcal{I}_i^a(t) = \{k \in \mathcal{I} \mid \alpha_k^{[i]}(t) = 0\}$ denotes the index set of all tasks that are considered *taken*. Initially we assume that $\alpha_k^{[i]}(t_0) = n_k$ for all agents $i$ and so $\mathcal{I}_i^a(t_0) = \mathcal{I}$ and $\mathcal{I}_i^t(t_0) = \emptyset$. Define, further, a *selection* function $f_s : 2^{\mathbb{R}} \to \mathbb{R}$ by $f_s(\mathcal{X}) \triangleq x \in \mathcal{X}$ for any set $\mathcal{X} \in \mathbb{R}$, where $x \in \mathcal{X}$ can be chosen according to any policy, deterministic or not. Applied on the index sets $\mathcal{I}_i^a(t)$, the selection function $f_s$ returns an available task $k \in \mathcal{I}_i^a(t)$ to which agent $i$ may be assigned.

Construction of the aforementioned discrete assignment component relies on distributed *market-based* coordination, where the agents are able to bid for a desired task assignment. In particular, we say that a task $k \in \mathcal{I}$ is being "sold" in the market if there is at least one agent $i$ such that $f_s(\mathcal{I}_i^a(t)) = k$. In the absence of a centralizing auctioneer, local communication among the agents is required to achieve the desired coordination. Let $\mathcal{N}_i(t)$ denote the set of neighbors of agent $i$, indicating those agents that agent $i$ can communicate with at time $t$. Each auction of the proposed market-based coordination scheme is described in Algorithm 1, while the assignment process consists of a sequence of such auctions taking place locally among neighboring groups of agents (Fig. 1).

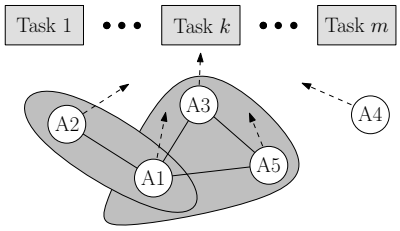



Fig. 1. *Forming local auctions*: Consider agents A1 through A5, all requesting to be assigned to Task $k$. Solid lines between the agents indicate communication links. In the proposed scenario, two *local* auctions are formed involving agents A1 and A2, and A1, A3 and A5, respectively. Note that A1 participates in both auctions. If $bid(A1) > \max\{bid(A2), bid(A3), bid(A5)\}$, then A2, A3 and A5 will switch to a different task, while A1 and A4 will keep requesting Task $k$. If $bid(A2) > bid(A1) > \max\{bid(A3), bid(A5)\}$, then A1, A3 and A5 will switch to a different task. Similarly, other bidding scenarios can be considered and the local auctions are guaranteed to break any ties over Task $k$.

Every auction takes place among an agent and all of its neighbors that wish to be assigned to the same task (line 1, Algorithm 1). The outcome of the auction depends on the values of the bids that the involved agents have placed. In particular, the agent with the highest bid wins the auction and continues to pursue its desired task (line 6, Algorithm 1). If there is a tie within the bids, the involved agents bid higher for the same task (line 10, Algorithm 1), while each agent with a lower bid selects a new task among the available ones (line 14, Algorithm 1) and updates its availability by a one unit decrement (line 15, Algorithm 1). In terms of the proposed market, this unit decrement indicates that the task is being "sold" in the market *one more time*. Note that this availability can never increase as the aforementioned task will always be eventually assigned to an agent (possibly not to the first one to claim an assignment) depending on the bidding process. In addition to decrementing a task's availability once an agent requests an assignment, every agent also updates task availabilities by setting them to the minimum value of itself and its neighbors (lines 3, 7 and 13, Algorithm 1). This minimum update rule is based on the fact that each time a task is "sold" in the market it will always be eventually assigned to an agent. Agents that are not aware of a task's availability can claim it (line 14, Algorithm 1) and eventually bid for it. The minimum update rule, however, informs agents about a tasks' availabilities and prohibits them from requesting assignments with unavailable tasks. This speeds up the assignment process, but more importantly, guarantees progress towards reaching a final assignment as it does not allow continuous bidding for any particular task, which could hinder convergence of the assignment algorithm. Note that in the case of $n$ tasks, each with availability $n_k = 1$, the proposed algorithm reduces to the one described in [15].

[1] We call task $k$ *unavailable* if it has been assigned to $n_k$ agents and *available* otherwise.

Correctness of the proposed scheme critically relies on the assumption that every agent requesting to be assigned to a task will eventually be able to communicate and compare bids with all other agents requesting to be assigned to the same task. In other words, it is assumed that every conflict on any particular task can be resolved as eventually all involved agents are able to communicate and collectively initiate a *local* auction and determine a final assignment, if required (Fig. 1).[2] Finally, the bidding process is assumed to be *sound*, namely every *tie* on the bids is eventually broken (line 9 of Algorithm 1). With these assumptions we can now state our main result.

*Proposition 3.1:* Given $n$ agents, each implementing Algorithm 1, the group is always guaranteed to reach an assignment that respects the availabilities of every task. Furthermore, the final assignment is reached after the group has explored at most $n(n+1)/2$ assignments.

*Proof:* Clearly, the minimum update rule and the assumption that all agents requesting to be assigned to a task will eventually be able to communicate and compare bids with all other agents requesting to be assigned to the same task, guarantee that the final assignment will respect the task availabilities. As discussed above, this is because all agents requesting to be assigned to a task will eventually share common availability information and will be able to initiate a final auction to determine the final assignment, if required. On the other hand, the assumption that any ties in the bids are always eventually broken guarantees that no auction can ever deadlock. This, combined with the fact that agents are prohibited from bidding for unavailable tasks (minimum update rule) guarantees progress towards the final assignment in finite time. To determine a bound on the number of assignments that need to be explored before the final assignment is reached we construct the following worst case scenario.

Assume $n$ tasks with availability one each and let the agents explore tasks sequentially, in a decreasing-bid order. Without loss of generality, we may assume that the agent with the highest bid gets assigned to the first task. Then, the second highest bid agent loses one auction for the first task and is assigned to the second task. The third highest bid agent loses two actions for the first and second task, respectively, and is assigned to the third task. Continuing in this way and summing up the number of times each agent had to pursue a new task gives a total number of assignments explored equal to $n(n+1)/2$. For more details on correctness of the proposed scheme, we refer the reader to [15]. ■

## IV. Experimental Setup and Software Design

The remainder of the paper is dedicated to verifying the effectiveness of the algorithm presented in Sect. III in the context of formation control of a team of robots. We begin by discussing implementation details relevant to the analysis and experimental design that follows.

[2] As an example, consider the case of a motion planning task where robots are required to *physically* visit targets and communication is proximity-based. Then, the aforementioned communication assumption is clearly satisfied.

**Algorithm 1** Market-Based Coordination for Agent $i$

**Require:** A task $k_i = f_s(\mathcal{I})$ and bid $b_i \geq 0$

1: Compute the set of agents bidding for the same task $k_i$, i.e., $\mathcal{A}_i := \{j \in \mathcal{N}_i \mid k_j = k_i\}$
2: **if** $\mathcal{A}_i = \emptyset$ **then**
3: Update the availabilities by, $\alpha_k^{[i]} := \min_{j\in\{i,\mathcal{N}_i\}}\{\alpha_k^{[j]}\}$ for all tasks $k \in \mathcal{I}$,
4: Update the sets $\mathcal{I}_i^a$, $\mathcal{I}_i^t$ and neighbors $\mathcal{N}_i$,
5: **else if** $\mathcal{A}_i \neq \emptyset$ **then**
6: **if** $b_i > \max_{j\in\mathcal{A}_i}\{b_j\}$ **then**
7: Update the availabilities by, $\alpha_k^{[i]} := \min_{j\in\{i,\mathcal{N}_i\}}\{\alpha_k^{[j]}\}$ for all tasks $k \in \mathcal{I}$,
8: Update the sets $\mathcal{I}_i^a$, $\mathcal{I}_i^t$ and neighbors $\mathcal{N}_i$,
9: **else if** $b_i = \max_{j\in\mathcal{A}_i}\{b_j\}$ **then**
10: Bid higher for the same task $k_i$, i.e., $b_i := f_s([b_i, \infty))$,
11: Update neighbors, $\mathcal{N}_i$,
12: **else if** $b_i < \max_{j\in\mathcal{A}_i}\{b_j\}$ **then**
13: Update the availabilities by, $\alpha_k^{[i]} := \min_{j\in\{i,\mathcal{N}_i\}}\{\alpha_k^{[j]}\}$ for all tasks $k \in \mathcal{I}$,
14: Select a new available task by, $k_i := f_s\big(\mathcal{I}_i^a \backslash \{k \mid \alpha_k^{[j]} = 0\}\big)$
15: Bid higher for the new task by, $b_i := f_s([b_i, \infty))$ and update its availability by $\alpha_{k_i}^{[i]} := \alpha_{k_i}^{[i]} - 1$,
16: Update the sets $\mathcal{I}_i^a$, $\mathcal{I}_i^t$ and neighbors $\mathcal{N}_i$,
17: **end if**
18: **end if**

### A. Algorithm

The distributed auctioning algorithm was implemented in *C++* using the open-source robotics software *Player*, part of the *Player/Stage/Gazebo* project [16]. The *Player* server enables network communications between multiple robots. *Player* also permits integration with simulation allowing the same code base to be used in both simulation and experimentation on the real hardware.

Each agent communicates a message packet containing its current position and item selection, availability estimate, and bid. The position is used to model limited communication neighborhoods in simulation and experimentation as well as to provide position estimates to the underlying control algorithms for local obstacle avoidance. Additionally, each agent chooses any initial or updated items from a random uniform distribution over the set of items with non-zero availability.

Implementing the algorithm on a team of robots introduces network delays. Additionally, because the system is distributed the resulting implementation is asynchronous. To alleviate these issues, the algorithm is divided into two modules which monitor the auction in multiple concurrent threads on each robot. An underlying message relay operating at high frequency aggregates and provides messages to the algorithm. The relay also tracks and prunes data below

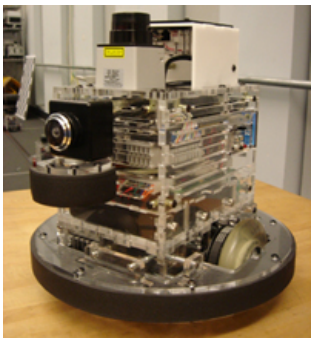


Fig. 2. *Scarab* Robot. The differential drive ground platform used in experimentation. Each robot is equipped with on-board computation, a tracking target for global pose estimation, and 802.11a wireless communications.

a threshold of "liveliness," to emulate the effects of lost network communications between neighbors.

### B. Simulation and Experimentation

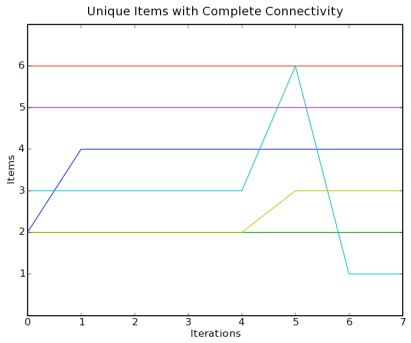
As noted in Sect. IV-A, the algorithm implemention uses the networking capabilities of *Player*. Simulations are performed using *Gazebo*, a 3D simulator that interfaces with *Player*. In simulation, *Player* uses shared memory to exchange messages locally. For this reason simulation results do not contain network delays. Additionally, *Player* currently only supports TCP communications and thus the network message passing is peer-to-peer rather than broadcast.

A team of six *Scarab* robots, one of which is shown in Fig. 2, serves as the experimental platform. The *Scarab* is a differential drive non-holonomic robot with on-board computation. Each *Scarab* is equipped with a localization target that provides accurate pose estimation in real time. Additionally, the *Scarabs* communicate on a wireless 802.11a network. Further details of the robot design and experimental environment are provided in [17].

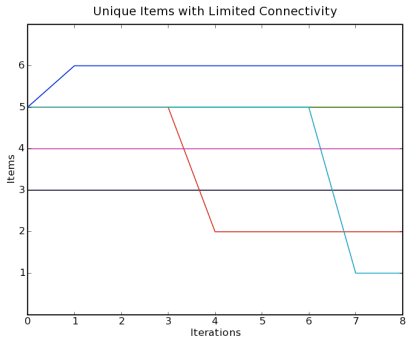
## V. Algorithm Performance

We tested the algorithm for effectiveness under a variety of conditions and discuss here two cases evaluated on a team of six robots with complete and limited network connectivity (see Sect. IV-A). By limited network connectivity, we mean that although all agents are connected, they do not all share the same set of neighbors. Each scenario was tested using two auction definitions: a unique item for each robot ($n_k = 1$, $k = \{1, \ldots, 6\}$) and two items shared equally between the six robots ($n_k = 3$, $k = \{1, 2\}$). The results of one trial auction for each of the four resulting test cases are shown in Fig. 3.

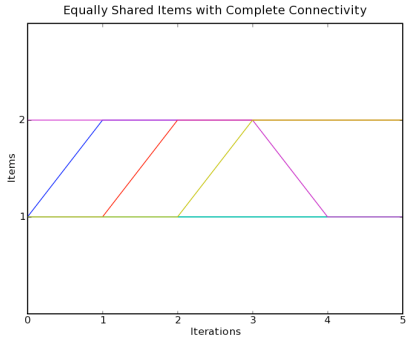
Figures 3(a) and 3(b) demonstrate the effects of complete versus limited communication on the update of the algorithm for two trial auctions. Clearly, Fig. 3(a) depicts an immediate response to any commonly shared items since information is globally available in a single iteration. However, when communication is limited (Fig. 3(b)) the two robots share a common item for several iterations before the information is dispersed to the other agent.



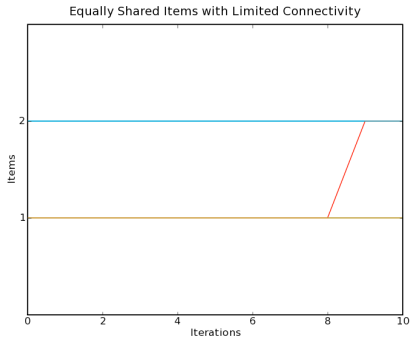

(a)




(b)




(c)




(d)

Fig. 3. Algorithm Performance. Four trials involving six robots demonstrate the effects of complete and limited inter-agent communication on auctions consisting of unique and equally shared items. Each line represents the current item selection of a robot. Note that since each auction is occurring locally and asynchronously, the $x$-axis depicts iterations rather than time.

Figures 3(c) and 3(d) show similar results. Empirically, we observe that in Fig. 3(d), although initially the auction is randomly divided such that it favors completion, a single agent shares an item that exceeds the permissible availability. Note that one would expect the robot to change on iteration seven since there are six agents in the system and full knowledge should therefore take six hops. However, as the robots are computing these values asynchronously and with network delays, it is reasonable to expect that some robots are able to perform faster updates and are therefore more advanced in the progression of the algorithm.

## VI. Application to Distributed Formation Control

In distributed robotics, the assignment problem addressed in Algorithm 1 naturally arises in applications involving destination and target allocation. The formation control problem is an example of one such application.

Consider the assignment of multiple tasks associated with predefined spatial locations or regions which define a formation. For the case where each robot is associated with a unique task $k$, the resulting auction definition is $n_k = 1$ for all $k$ and $m = n$ under the assumptions discussed in Sect. III. Such a situation resolves to a unique point and can therefore be considered as distributed formation control. However, if $n_k > 1$, then the resulting auction dynamically groups a subset of robots. In such a situation, $k$ is a shared item or task, which we choose to associate with a region in space. The resulting auction therefore defines group splitting and merging depending upon the auction specification.

Both of these auctions require that an underlying control law exists concurrently with the capability to resolve inter-agent collision avoidance and way-point navigation. Of course, any control law suitable for the application is acceptable. For the purposes of our implementation, we have selected a control law modeled closely after the distributed navigation function controllers presented in [18], [19] for nonholonomic, non-point robots. Reactive control laws are also permissible. Recall that each agent is transmitting its current position within the message packet described in Sect. IV-A. This information permits a control law to resolve collision scenarios without requiring further information.

### A. Simulation Results

We tested Algorithm 1 in a distributed formation control scenario where robots are uniquely assigned to spatial points in $\mathbb{R}^2$ that collectively spell the name "GRASP." A simulated team of twenty non-holonomic robots modeled closely to agree with the physical characteristics of the *Scarab* robot perform way-point navigation, while avoiding collisions, to a location dynamically assigned by the distributed auction. A thresholding of messages originating more than $5\,\mathrm{m}$ from each robot models the effect of limited communication on the algorithm. The series of images in Fig. 4 provides a depiction of the evolution of the simulation.

### B. Experimental Results

We experimentally tested the algorithm on six *Scarab* robots. In Sect. VI-A, we consider the formation problem where each robot is bidding for a unique location. In experimentation, we consider the capabilities of the algorithm to resolve auctions involving shared items where each item represents a region characterized by a disc centered at a specific location in the experimental environment.

From an initial formation (Fig. 5(a)) the robots are transmitted a series of auction descriptions that defined the number of items being auctioned as well as the maximum availabilities. The underlying code base ensures that message queues are cleared between auctions to prevent old messages from effecting the next manually triggered auction.

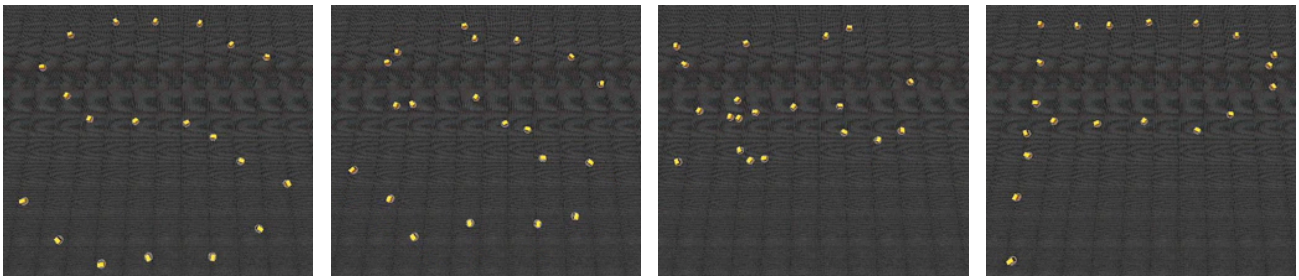
An auction defined by $m = 2, n_1 = n_2 = 3$, determines the formation shown in Fig. 5(b), where $k_1$ and $k_2$ correspond to the points $(0, -1.75\,\mathrm{m})$ and $(0, 1.75\,\mathrm{m})$ in the global reference frame, respectively. The resulting control specification causes a division of the group into two subgroups of equal size. Figure 5(c) depicts the formation defined by a similar auction transmitted to the robots, but with a specification that further splits the team into four subgroups. The auction is defined by $m = 4$, $n_1 = n_2 = 1$, $n_3 = n_4 = 2$, where tasks $k_1$ and $k_2$ were defined previously and $k_3$ and $k_4$ correspond to the points $(1.75\,\mathrm{m}, 0)$ and $(-1.75\,\mathrm{m}, 0)$.

## VII. Discussion and Future Work

We addressed the problem of distributed task allocation for a team of robots and proposed a distributed algorithm that requires at most $n(n+1)/2$ iterations to converge. The general definition of tasks or items makes this algorithm applicable to a wide range of scenarios. We discussed the performance and implementation of the algorithm on a team of robots. We concluded with simulation and experimental results that demonstrated the effectiveness of the algorithm for applications such as distributed formation control and merging and splitting of groups.

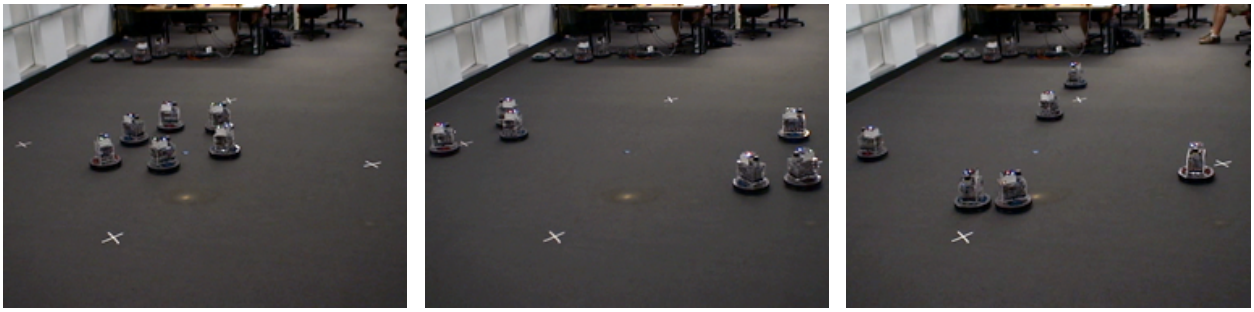
A future study on the effect of broadcast versus peer-to-peer communication in an ad-hoc network for a larger system is merited. In this paper, we modeled limited communication by ignoring messages from agents that are not within a predefined neighborhood due to a limited experimental area. Additionally, we characterized the algorithm with six robots using TCP communications.

Other future directions include the application of the algorithm to tasks such as distributed manipulation, pattern generation, persistent surveillance, and environmental monitoring, where the ability to spilt or merge a team of robots depending upon object or pattern shape or the geometry of the environment becomes relevant. Finally, an integration of the algorithm with mixed initiative control, where tasks are allocated concurrently by a human or supervisory agent, would permit interaction with only a subset of the team of robots while the remaining robots dynamically adjust to the changing system.



(a) (b) (c) (d)

Fig. 4. Simulated Formation Control. Twenty nonholonomic robots modeled after the *Scarab* control to a series of letters forming the local laboratory name "GRASP." Each robot has a limited sensing and communication range. Concurrent with the auction, each robot is performing decentralized collision avoidance and way-point control. Figures 4(a)–4(d) depict the transition between the two letters "S" and "P." The full simulation is captured in the supplemental video.



(a) (b) (c)

Fig. 5. Experimental Results. A team of six *Scarab* robots perform formation control defined by distributed task assignment. The robots begin as a group in Fig. 5(a). The team is split evenly between two regions in Fig. 5(b). Four regions define the final auction and formation control shown in Fig. 5(c). Each region is marked by a white "X."

## REFERENCES


[1] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[2] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sept. 2004.

[3] H. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 863–868, May 2007.

[4] J. Cortes, S. Martinez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1289–1298, Aug. 2006.

[5] R. Sepulchre, D. Paley, and N. E. Leonard, "Stabilization of planar collective motion: All-to-all communication," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 811–824, May 2007.

[6] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA, May 2004, pp. 165–172.

[7] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics*, vol. 2, no. 1, pp. 83–97, 1955.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W. H. Freeman, 1979.

[9] H. A. Almohamad and S. O. Duffuaa, "A linear programming approach for the weighted graph matching problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 5, pp. 522–525, May 1993.

[10] M. M. Zavlanos and G. J. Pappas, "A dynamical systems approach to weighted graph matching," in *Proc. of the IEEE Conference on Decision and Control*, San Diego, CA, Dec. 2006, pp. 3492–3497.

[11] K. Lerman, C. Jones, A. Galstyan, and M. J. Mataric, "Analysis of dynamic task allocation in multi-robot systems," *The Int. Journal of Robotics Research*, vol. 25, no. 3, pp. 225–242, Mar. 2006.

[12] M. M. Zavlanos and G. J. Pappas, "Dynamic assignment in distributed motion planning with local coordination," *IEEE Transactions on Robotics*, vol. 24, no. 1, Feb. 2008.

[13] M. Ji, S. Azuma, and M. Egerstedt, "Role-assignment in multi-agent coordination," in *Int. Journal of Assistive Robotics and Mechatronics*, vol. 7, no. 1, Mar. 2006, pp. 32–40.

[14] S. L. Smith and F. Bullo, "Target assignment for robotic networks: Asymptotic performance under limited communication," in *Proc. of the American Control Conf.*, New York, July 2007, pp. 1155–1160.

[15] M. M. Zavlanos and G. J. Pappas, "Distributed formation control with permutation symmetries," in *Proc. of the IEEE Conference on Decision and Control*, New Orleans, LA, Dec. 2007, pp. 2894–2899.

[16] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. of the Int. Conf. on Advanced Robotics*, Coimbra, Portugal, June 2003, pp. 317–323.

[17] N. Michael, J. Fink, and V. Kumar, "Experimental testbed for large multi-robot teams: Verification and validation," *IEEE Robotics and Automation Magazine*, Mar. 2008.

[18] D. V. Dimarogonas and K. J. Kyriakopoulos, "A feedback stabilization and collision avoidance scheme for multiple nonholonomic non-point agents," in *Proc. of IEEE Int. Symposium on Computational Intelligent Control*, Limassol, Cyprus, June 2005, pp. 820–825.

[19] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos, and M. M. Zavlanos, "A feedback stabilization and collision avoidance scheme for multiple independent non-point agents," *Automatica*, vol. 42, no. 2, pp. 229–243, Dec. 2006.