Centralized and Distributed Task Allocation in Multi-Robot Teams via a Stochastic Clustering Auction

KAI ZHANG
Florida A&M University-Florida State University
EMMANUEL G. COLLINS
Florida A&M University-Florida State University
and DONGQING SHI
Dartmouth College

This paper considers the problem of optimal task allocation for heterogeneous teams, e.g., teams of heterogeneous robots or human-robot teams. It is well known that this problem is NP hard and hence computationally feasible approaches must develop an approximate solution. Here, we propose a solution via a Stochastic Clustering Auction (SCA) that uses a Markov Chain search process along with simulated annealing. This is the first stochastic auction method used in conjunction with global optimization. It is based on stochastic transfer and swap moves between the clusters of tasks assigned to the various robots and considers not only downhill movements, but also uphill movements, which can avoid local minima. A novel feature of this algorithm is that, by tuning the annealing suite and turning the uphill movements on and off, the global team performance after algorithm convergence can slide in the region between the global optimal performance and the performance associated with a random allocation. Extensive numerical experiments are used to evaluate the performance of SCA in terms of costs and computational and communication requirements. For centralized auctioning, the SCA algorithm is compared to fast greedy auction algorithms. Distributed auctioning is then compared with centralized SCA.

Categories and Subject Descriptors: I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Coherence and coordination; G.2.1 [Discrete Mathematics]: Combinatorics—Combinatorial algorithms; Permutations and combinations; I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Graph and tree search strategies; Heuristic methods; G.3 [Probability and Statistics]: Probabilistic algorithms (including Monte Carlo); I.2.9 [Artificial Intelligence]: Robotics; I.2.1 [Artificial Intelligence]: Applications and Expert Systems—Games

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Auctions and market-based systems, optimal task allocation, distributed robot systems, networked agents, Markov Chain Monte Carlo, simulated annealing

Author’s address: K. Zhang and E. Collins are with Center for Intelligent Systems, Control and Robotics (CISCOR), Department of Mechanical Engineering, FAMU-FSU College of Engineering, Florida A&M University-Florida State University, Tallahassee, FL 32310.
D. Shi is with Department of Psychological and Brain Sciences, Dartmouth College, 6207 Moore Hall, Hanover, NH 03755.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 2010 ACM 0000-0000/2010/0000-0001 $5.00
1. INTRODUCTION

Research on coordinating heterogeneous robots has come into prominence due to the anticipated demand for teams that consist of both heterogeneous unmanned and manned vehicles for both civil and military applications. A critical issue is how to assign tasks to these heterogeneous robots in order to optimally or near-optimally complete a given mission. It is well known that this task allocation must not always rely on complete communication between the allocator and each of the robots. Hence, a practically meaningful solution must allow distributed task allocation. In addition, for the scenario considered above, the presence of heterogeneous vehicles and humans introduces additional factors that must be considered when allocating tasks. In particular, differing robot mobility, and differing task completion times for the heterogeneous robots and humans must be taken into account.

Existing approaches to task allocation can be placed into five categories: 1) fully centralized approaches, 2) centralized auctions, 3) distributed auctions, 4) completely distributed approaches, and 5) hybrid approaches of distributed auctions and emergent coordination.

Fully centralized approaches [Chao et al. 1993; Fredrickson et al. 1978; Koes et al. 2006] require a central allocator to determine the task distribution for the team based on the allocator’s model of each member of the team, which enables the allocator to compute the cost of completing a given task for each team member. These methods seek to determine the task allocation to optimize a global cost function. Since this problem is NP hard [Gerkey and Mataric 2004], a variety of heuristic optimization algorithms are employed. This optimization of the global cost function is the primary advantage of these approaches. However, there are three important disadvantages. First and foremost, fully centralized approaches require complete communication between the central allocator and each team member, which is not always feasible. Second, they require the central allocator to have substantial internal knowledge of each team member and to keep track of changes in the team members as the mission progresses, which may not be practical. Third, they are vulnerable to a single point failure.

Centralized auctions involve a central auctioneer that determines the task allocation based on the task bids provided to it by each team member. The task allocation can be accomplished via one of several auctioning methods, such as a combinatorial auction [Cramton et al. 2006; de Vries and Vohra 2003; Hoos and Boutilier 2000; Koenig et al. 2006; Mito and Fujita 2004; Sandholm 2002] or a greedy auction [Koenig et al. 2006]. Due to the distribution of the cost computations for each robot, this method does not require the central auctioneer to have and keep track of an internal model of each team member. However, it has the communication and computational disadvantages of the fully centralized approaches. It does share the advantage of providing a framework for considering optimization of a global cost function. Though combinatorial auctioning [Cramton et al. 2006; de Vries and Vohra 2003; Koenig et al. 2006; Mito and Fujita 2004; Sandholm 2002] provides global optimal solutions, it is well known to be an exponential algorithm [Sandholm 2002] and thus becomes practically infeasible as the number of tasks and robots increase. Hence, the work of [Sandholm 2002] uses deterministic heuristics to limit the combinations considered by greedily constructing the bid tree, while the work
of [Hoos and Boutilier 2000] integrates random walk into the greedy construction of the bid tree.

Distributed auctions [Andersson and Sandholm 2000; Botelho and Alami 1999; Brunet 2008; Choi et al. 2008; 2009; Clark et al. 2008; Dias and Stentz 2002; Dias et al. 2006; Gerkey and Mataric 2002; Golfarelli et al. 1997; Koenig et al. 2007; McLurkin and Yamins 2005; Sandholm 1998; Zheng and Koenig 2009; Zlot and Stentz 2006] involve peer-to-peer redistribution of plans between given subsets of robots, where one of the robots serves as the auctioneer. This class of methods is the primary mechanism for providing intentional coordination between robots. Like central auctions they have the advantage of not requiring the auctioneer to have a model of each team member. They also do not require full communication between the robots and the auctioneer and can be robust with respect to communication failures. However, they are inherently suboptimal when the global cost function is considered.

It should be noted that, in general, centralized auction approaches lead to distributed auction approaches using the concept of “opportunistic centralization” [Dias and Stentz 2002], such that the centralized auction algorithm is applied regionally. This concept is used in this paper in Section 2.3. Opportunistic centralization is inherent in all of the distributed auction approaches. (One way to see this is that each of the distributed auction methods corresponds to a centralized auction method when the auction simultaneously involves each of the robots, i.e., a regional auction is the global auction.)

Distributed auctions can generally be divided into three classes. The first set [Botelho and Alami 1999; Clark et al. 2008; Gerkey and Mataric 2002; Golfarelli et al. 1997; McLurkin and Yamins 2005; Zlot and Stentz 2006] uses greedy auctioning, which is inherently suboptimal. The second set [Brunet 2008; Choi et al. 2008; 2009; Dias and Stentz 2002; Dias et al. 2006; Koenig et al. 2007] uses the same deterministic heuristics as in [Sandholm 2002] to limit the combinations considered in combinatorial auctioning. A current limitation of these methods is that the deterministic heuristics assume that the triangle inequality is preserved for the metric cost space [Aarts and Lenstra 1997], which does not apply to cost functions that can be used to represent minimum time objectives (see (7) below). The third set of auction methods [Andersson and Sandholm 2000; Sandholm 1998; Zheng and Koenig 2009] is closely related to the method developed here. This set uses a deterministic synthesis of single-transfer, swap and multi-party exchange movements between the clusters assigned to the robots. However, a current common limitation of all the approaches among these three classes is that they do not provide a mechanism to avoid local minima [Andersson and Sandholm 2000; Aarts and Lenstra 1997] (see Section 2.2.2).

An additional limitation of the previously developed auction methods is that they do not provide a mechanism for using computational and communication requirements to enable the performance obtained after the algorithm convergence can slide in the region between the globally optimal performance and the performance associated with some random allocation as illustrated in Fig. 1. In particular, once these algorithms converge for a given problem they converge to a single cost. However, it may be desirable to specify that one is willing to increase (or decrease)
Fig. 1. Illustration of the ability of an “ideal auction method” to trade off computational and communication requirements so that the converged performance lies anywhere in the performance spectrum.

computational and communication requirements in order to increase (or decrease) the allocation performance by decreasing (or increasing) the converged cost.

**Fully distributed approaches** [Parker 1998; Stroupe 2003; Wagner and Arkin 2004] do not require direct communication between the robots. Each robot retains local control and chooses its own actions based on its observations of the environment. For example, in learning approaches, particularly reinforcement learning [Köse et al. 2004; Sutton and Barto 1998], the self-interested robots are usually fully distributed. Cooperative actions, however, can be achieved by estimating models of the other robots’ strategies [Powers and Shoham 2005] or by applying heuristics [Bowling and Veloso 2002]. Distributed approaches are obviously robust with respect to communication failures, but can yield more suboptimal task allocation than distributed auctions due to lack of intentional coordination via an auctioneer.

**Hybrid approaches of distributed auctions and emergent coordination** [Jones et al. 2006; Simmons et al. 2007] incorporate implicit negotiation (i.e., emergent coordination) into a larger framework of explicit negotiation (i.e., intentional coordination) to obtain some of the benefits of both types of coordination. Due to the nature of distribution, however, these methods can yield highly suboptimal task allocation.

This research is fundamentally concerned with contributing to the rich literature on centralized and distributed auction methods by developing an algorithm that is applicable to a variety of cost functions, has the ability to avoid local minima, and has transparent mechanisms to slide in the region shown in Fig. 1 between the global optimal performance and the performance of a random allocation. Sandholm has suggested that probabilistic or stochastic algorithms may result in better approximation algorithms [Sandholm 2002]. The algorithm developed here is the first stochastic algorithm for task allocation using global optimization and is in the class of Markov Chain Monte Carlo methods [Robert and Casella 2005]. The optimization mechanism is simulated annealing [Kirkpatrick et al. 1983]. The algorithm is called a Stochastic Clustering Auction (SCA) and was initially proposed in [Zhang et al. 2008] for centralized and decentralized auctions. It is based on recent algorithms that were successfully used for shape clustering and segmentation in computer vision [Barbu and Zhu 2005; Srivastava et al. 2005]. SCA alternates with
equal probabilities between transfer and swap moves and allows not only downhill movements, but also uphill movements, which can enable it to avoid local minima. By tuning the annealing suite and turning the uphill movements on and off, the team performance obtained after algorithm convergence can slide in the region between the global optimal performance and the performance of a random allocation. The developments are for heterogeneous robots (see Section 2.2.3 and 3.1.1).

Extensive simulations are used to evaluate the performance of SCA in random scenarios. Particular attention is given to a comparison of the performance of SCA with that of standard greedy auction algorithms, notably the sequential auction, the parallel auction, and two of their variants, described in Appendix B. It is also shown how the type of random simulations presented here can provide guideline information for choosing the number of robots needed in a given mission for an expected number of tasks.

The remainder of this paper is organized as follows. Section 2 formulates the basic optimization problem for task allocation, provides a description of a SCA algorithm, and discusses how the algorithm may be used for both centralized and distributed auctioning. Section 3 considers centralized auctioning and presents simulation results from random scenarios with a focus on a comparison of the results of the SCA algorithm with and without uphill movements and a comparison of the SCA algorithms with standard greedy algorithms. Section 4 considers distributed auctioning and presents simulation results from random scenarios and for selected benchmark auction patterns with a focus on a comparison of the performance achieved with distributed and centralized SCA. Finally, Section 5 presents conclusions and future work.

2. STOCHASTIC CLUSTERING AUCTION FOR CENTRALIZED AND DISTRIBUTED TASK ALLOCATION

This section first presents the basic problem statement. It then describes the Stochastic Clustering Auction (SCA). After introducing the concept of regional cost, it is shown that when a distributed auctioneer reduces the corresponding regional cost, the global cost will either decrease or remain the same. Hence SCA is proposed to optimize the regional cost in a distributed auction.

2.1 Problem Statement

Let \( \mathcal{H} \) denote a set of \( k \) heterogeneous robots, and \( \mathcal{T} \) denote a set of \( n \) tasks, i.e.

\[
\mathcal{H} = \{ h_1, h_2, \ldots, h_k \}, \quad (1)
\]
\[
\mathcal{T} = \{ t_1, t_2, \ldots, t_n \}. \quad (2)
\]

Also, let \( \mathcal{A} \) denote the allocation,

\[
\mathcal{A} = \{ a_1, a_2, \ldots, a_k \}, \quad (3)
\]

where \( a_s \) is a cluster of tasks,

\[
\bigcup_{s=1}^{k} a_s = \mathcal{T}, \quad (4)
\]
\[
a_s \cap a_t = \emptyset, (s \neq t). \quad (5)
\]
and the cluster \( a_s \) is assigned to robot \( h_s \). The cost associated with \( \mathcal{A} \) is given by either

\[
C(\mathcal{A}) = \sum_{s=1}^{k} c_s(a_s),
\]

or

\[
C(\mathcal{A}) = \max_{s} c_s(a_s),
\]

where \( c_s(a_s) \) is the minimum cost for robot \( h_s \) to complete the set of tasks \( a_s \). The individual cost function \( c_s(\cdot) \) is based on characteristics of each robot, e.g. the dynamic model of the robot, the state of the market, and current task commitments. The problem is to solve the optimization

\[
\min_{\mathcal{A}} C(\mathcal{A}).
\]

In practice the cost function in (6) might be used to represent the total distance traveled or the total energy expended by the robots while the cost function in (7) might be used to represent the maximum time taken to accomplish the tasks or the mission length.

2.2 Stochastic Clustering Auction (SCA)

In the auctioning framework [Dias et al. 2006], SCA attempts to minimize the cost \( C(\mathcal{A}) \) using a Markov chain search process in the space of possible allocations. It is assumed that the robots are cooperative, and that collusion, shilling and other cheating mechanisms are not allowed [Dias et al. 2006]. The basic algorithm was originally developed in [Srivastava et al. 2005]. The algorithm is a Gibbs sampler in the class called Markov Chain Monte Carlo [Robert and Casella 2005]. The essential mechanism of SCA is to start with an allocation \( \mathcal{A} \) for \( k \) clusters and to reduce or probabilistically hillclimb \( C(\mathcal{A}) \) by rearranging the tasks \( T \) among the clusters. The rearrangement is performed in a stochastic fashion using transfer and swap moves. These moves are performed with probabilities proportional to the negative exponential of the costs \( C(\mathcal{A}) \) of the resulting allocations \( \mathcal{A} \) (see (9) and (10)). SCA is always guaranteed to result in an allocation that has a cost less than or equal to the cost of the initial allocation. The actual algorithm is described below.

2.2.1 Stochastic Clustering Auction

(1) The auctioneer partitions \( T \) into \( k \) clusters to form an initial allocation \( \mathcal{A}^{(0)} = \{a_1^{(0)}, a_2^{(0)}, \ldots, a_k^{(0)}\} \), where each cluster \( a_j^{(0)} \) is an unordered subset of \( T \). Let \( \mathcal{A} = \mathcal{A}^{(0)} \) and \( \mathcal{A}^* = \mathcal{A}^{(0)} \). (\( \mathcal{A} \) is the current algorithm allocation, while \( \mathcal{A}^* \) is the allocation that has the lowest cost.)

(2) Each robot \( h_p \in \mathcal{H} \ (p = 1, 2, \ldots, k) \) uses a “constrained Prim’s Algorithm”\(^1\) (a greedy algorithm) to efficiently approximate the cost \( c_p(a_p) \) and submits its cost to the auctioneer. In this bid valuation stage, each cluster \( a_p \) becomes an

\(^1\)This algorithm fixes the initial vertex with a single edge in Prim’s Algorithm [Jarnick 1930; Prim 1957] as building a minimum spanning tree, and hence, unlike Prim’s algorithm, is not guaranteed to be optimal. It is well known to be 2-approximate (i.e., solution is at least \( \frac{1}{2} \) as good as optimal).
ordered subset of $\mathcal{T}$. The auctioneer computes the global cost $C(\mathcal{A})$ using (6) or (7) and sets a high temperature $T$.

(3) The auctioneer rearranges the clusters with equal probabilities selecting either a single move (a) or a dual move (b) after randomly selecting two clusters $a_s$ and $a_t$ for task transfers or swaps.

(a) **Single Move (Task Transfer):** Randomly select a task $t_i \in a_s$ from robot $h_s$. Assume that $t_i$ is reassigned to $a_p$ for robot $h_p$ ($p = 1, 2, \ldots, k$, with $p \neq s$), resulting in the new allocation $\mathcal{A}_i^{(s,p)}$ that has two modified clusters $a_s^{-i}$ and $a_p^{+(i)}$. Assume that robot $h_s$ computes $c_s(a_s^{-i})$ and robot $h_p$ computes $c_p(a_p^{+(i)})$, which the auctioneer uses to compute the corresponding cost $C(\mathcal{A}_i^{(s,p)})$ (based on (6) or (7)). The probability of the acceptance of the transfer of task $t_i$ from robot $h_s$ to robot $h_t$ is given by

$$P_S(i, s, t, T) = \frac{\exp(-C(\mathcal{A}_i^{(s,t)})/T)}{\sum_{p=1, p \neq s} \exp(-C(\mathcal{A}_i^{(s,p)})/T)}.$$  

If $C(\mathcal{A}_i^{(s,t)})$ is less than $C(\mathcal{A}^*)$, then the new $\mathcal{A}^*$ is updated to $\mathcal{A}_i^{(s,t)}$.

(b) **Dual Move (Task Swap):** Randomly select two tasks in $a_s$ and $a_t$, task $t_i$ from robot $h_s$ and task $t_j$ from robot $h_t$, and swap them, resulting in the new allocation $\mathcal{A}_{i,j}^{(s,t)}$ that has two modified clusters $a_s^{-(i+j)}$ and $a_t^{+(i-j)}$. Assume that robot $h_s$ computes $c_s(a_s^{-(i+j)})$ and robot $h_t$ computes $c_t(a_t^{+(i-j)})$, which the auctioneer uses to compute the corresponding cost $C(\mathcal{A}_{i,j}^{(s,t)})$ (based on (6) or (7)). Then, the probability of swapping the two tasks is given by

$$P_D(i, j, s, t, T) = \frac{\exp(-C(\mathcal{A}_{i,j}^{(s,t)})/T)}{\exp(-C(\mathcal{A}_{i,j}^{(s,t)})/T) + \exp(-C(\mathcal{A})/T)}.$$  

If $C(\mathcal{A}_{i,j}^{(s,t)})$ is less than $C(\mathcal{A}^*)$, then the new $\mathcal{A}^*$ is updated to $\mathcal{A}_{i,j}^{(s,t)}$.

(4) The auctioneer accepts the proposal with probability $P_S(i, s, t, T)$ or $P_D(i, j, s, t, T)$ so that $\mathcal{A}$ is updated and the cost $C(\mathcal{A})$ is put on log. Otherwise, the auctioneer declines the proposal and the auctioneer keeps the current configuration and goes back to Step (3).

(5) If the auction evolution termination criterion is satisfied, i.e., $T < T_{cut}$, where $T_{cut}$ is some threshold temperature, then the auction is terminated and the final allocation is $\mathcal{A}^*$ with final cost $C(\mathcal{A}^*) \leq C(\mathcal{A}^{(0)})$. If the criterion is not satisfied, reduce $T$, using $T \leftarrow T/\beta$ where $\beta > 1$ and go to Step (3).

During each cost evaluation of the bid evaluation cycle in Step (3) of the above SCA only two clusters are rearranged: the clusters for robots $h_s$ and $h_p$ in Single Move, and the clusters for robots $h_s$ and $h_t$ in Dual Move. Thus, there are $k - 2$

---

2 Each cluster is treated as an unordered subset and is ordered in a later bid valuation stage.

3 This cost is computed using the constrained Prim’s algorithm during bid valuation stages.
clusters unchanged. In order to achieve efficiency of SCA with respect to computational and communication cost, an inheritance calculation is used to calculate $C(A_i^{(s,p)})$ or $C(A_i^{(s,t)})$. Therefore, the cost after moving, is calculated for the cost defined in (6) as

$$C(A_i^{(s,p)}) = C(A) + (c_s(a^{(-i)}_s) - c_s(a_s)) + (c_p(a^{(+i)}_p) - c_p(a_p))$$

Single Move

$$C(A_i^{(s,t)}) = C(A) + (c_s(a^{(-i,+j)}_s) - c_s(a_s)) + (c_t(a^{(+i,-j)}_t) - c_t(a_t))$$

Dual Move

(11)

and for the cost in (7) as

$$C(A_i^{(s,p)}) = \max\{C(A), c_s(a^{(-i)}_s), c_p(a^{(+i)}_p)\}$$

Single Move

$$C(A_i^{(s,t)}) = \max\{C(A), c_s(a^{(-i,+j)}_s), c_t(a^{(+i,-j)}_t)\}$$

Dual Move

(12)

Note that in (11) $c_s(a^{(-i)}_s)$ and $c_p(a^{(+i)}_p)$ are computed in parallel while in (12) $c_s(a^{(-i,+j)}_s)$ and $c_t(a^{(+i,-j)}_t)$ are computed in parallel because in both cases the two costs are computed by different robots.

2.2.2 Further Discussion of SCA. Additional types of moves such as multi-party exchange or some combination of transfer, swap, and multi-party exchange can also be used to improve the search over the task space [Andersson and Sandholm 2000; Sandholm 1998; Zheng and Koenig 2009], although their computational cost becomes a factor too. In view of the computational simplicity of single transfer and swap tasks, we have restricted our algorithm to these two moves in Section 2.2.1. In the implementation of SCA used in this study, the algorithm alternates with equal probabilities between single and dual moves. Simulation results showed that when SCA alternates with equal probabilities between single and dual moves it is more efficient than using exclusively single moves or dual moves.

In order to search for the global optimum, a simulated annealing method has been adopted. Similar to the seminal work in [Kirkpatrick et al. 1983], a SCA starts with a high value of $T$ and gradually reduces it in order to to make small variations in the task allocation while searching for the optimal allocation in $T$. Although simulated annealing and the random nature of the search help in avoiding local minima, the convergence to a global minimum is difficult to establish. As described in [Robert and Casella 2005], the output of this algorithm is a Markov chain that is neither homogeneous nor convergent to a stationary chain. If the temperature $T$ is decreased slowly, then the chain is guaranteed to converge to a global minimum. Note that the use of an internal greedy algorithm (see Step (2) in Section 2.2.1) is likely to prevent the computation from converging to desired global optimum even if the annealing procedure converges. Hence, the primary purpose of using simulated annealing is to enable the algorithm to yield high-performance solutions with reasonably fast execution times, rather than guarantee asymptotic convergence to a global optimum.
Algorithm 1 principal mechanisms for cSCA and gcSCA

1: repeat
2:   Initialize.
3:   Recluster.
4:   Decide whether to accept the proposed cluster.
5:   if the solution is better (for gcSCA only) then
6:     Accept.
7:   else
8:     Accept with an acceptance probability.
9:   end if
10: until The termination is reached

2.2.3 Description of greedy centralized Stochastic Clustering Auction (gcSCA).
As mentioned in Section 2.2.1, centralized SCA (cSCA) is based on a Gibbs Sampler. cSCA may also be based on what is called here a Greedy Gibbs Sampler, resulting in the greedy cSCA (gcSCA). The principal mechanisms for cSCA and gcSCA are illustrated in Algorithm 1. Note that line 6 is valid only for cSCA and this uphill movement is turned off for gcSCA, which differentiates the two algorithms.

2.3 Use of SCA for Distributed Task Allocation
Since cSCA and gcSCA always reduce or hold the cost constant (see Step (6) of Section 2.2.1), motivated by Proposition 1, distributed SCA simply applies cSCA or gcSCA regionally. Details are referred to Appendix A.

3. EXPERIMENTAL RESULTS FOR CENTRALIZED IMPLEMENTATION OF SCA
This section provides simulation results for SCA using the multi-robot routing problem, which is a standard test domain for robot coordination using auctions [Koenig et al. 2006; Andersson and Sandholm 2000; Brunet 2008; Choi et al. 2008; 2009; Clark et al. 2008; Dias and Stentz 2002; Dias et al. 2006; Koenig et al. 2007; Sandholm 1998; Zheng and Koenig 2009]. The task allocation is time-extended assignment such that all tasks are assigned to robots before the assignments are carried out [Gerkey and Matarić 2004]. It is free of conflicts since each task is assigned to no more than one robot. The tasks in the multi-robot routing problem considered here are to visit targets and complete an assignment. In the simulations robot heterogeneity was taken into account by assuming that the robots moved at differing speeds and differed in their completion times at each target. The different completion times at each target have the effect of forcing different robots to play different roles. For example, if one robot can complete a task in 0.5 s and another robot requires 2000 s, SCA will almost certainly not assign the second robot to this task. The SCA task allocations are compared with those obtained using SA, LBSA, PA, and LBPA, which were described in Appendix B. For each simulation the stochastic random scenario appears in a 10000 m × 10000 m area. The initial robot positions were evenly distributed along one edge of the area. The speeds for each of the robots were assumed to be constant and were chosen randomly from the interval (0 m/s, 20 m/s) assuming a uniform distribution. Considering each robot, for each simulation the completion time for a given task was chosen with equal probabilities
among the three intervals of \((0,2000]\), \((0,20000]\), and \((0,200000]\), and then chosen randomly from the selected interval, assuming a uniform distribution.

The cost function is a \(\text{MinMAX}\) cost function in (7) corresponding to the mission completion time. (Similar results using the \(\text{MinSUM}\) cost function in (6) are omitted for brevity. However, a benchmark problem using the \(\text{MinSUM}\) cost function was presented in [Zhang et al. 2008].) Also, for each simulation the following SCA parameters were used: initial temperature, \(T = 1000\); and termination temperature, \(T_{\text{cut}} = 20\).

**Definition 1** The communication complexity of SCA is measured by the number of Auction Cycles (ACs). Formally, an AC is one bid evaluation cycle corresponding to Steps (3)-(4) of Section 2.2.1 for SCA.

In addition, to evaluate the performance of a SCA the concept of Mean Cost Improvement (MCI) is introduced as given by Definition 2.

**Definition 2** For \(m\) stochastic scenarios let \(\{C_{\text{SCA}}^r : r = 1, \ldots, m\}\) denote the set of \(m\) costs resulting from SCA and let \(\{C_{\text{BestGreedy}}^r : r = 1, \ldots, m\}\) denote the set of minimum costs achievable with the greedy algorithms: SA, LBSA, PA and LBPA. The Mean Cost Improvement (MCI) is the average of the normalized improvement of the SCA cost over the best of the greedy algorithms, such that

\[
MCI = \frac{1}{m} \sum_{r=1}^{m} \frac{C_{\text{BestGreedy}}^r - C_{\text{SCA}}^r}{C_{\text{BestGreedy}}^r}.
\]

This section studies the performance of cSCA, gcSCA, and the four greedy auction algorithms using simulations involving 1000 random scenarios for a given number of tasks and robots.

### 3.1 Comparison of cSCA, gcSCA, and the Greedy Auction Algorithms with 3 Robots

The initial simulations were restricted to 3 robots with the number of tasks ranging from 5 to 100 in increments of 5.

Table I. Comparison of the mean computational times of SCA with the ones of the four greedy auction methods for the stochastic scenarios involving 3 robots (SA and LBSA are omitted since PA and LBPA always outperform them in terms of both costs and computational times for \(\text{MinMAX}\) problems. For simplicity, the number of tasks ranging from 5 to 95 in increments of 15 is shown.)

<table>
<thead>
<tr>
<th>tasks</th>
<th>(\beta = 1.001)</th>
<th>(\beta = 1.01)</th>
<th>(\beta = 1.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(c\text{SCA})</td>
<td>(g\text{cSCA})</td>
<td>(c\text{SCA})</td>
</tr>
<tr>
<td>5</td>
<td>0.0027 0.0032 1.0032 0.1148 0.1098 0.0248 0.0123 0.008 0.0123 0.008 0.0123 0.008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.0182 0.0201 6.068 5.8149 6.075 5.904 0.066 0.0642</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>0.0469 0.051 12.5247 12.2543 1.2372 1.2 0.1322 0.127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.0868 0.0928 20.2669 20.0581 1.9922 1.9373 0.2143 0.2085</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>0.1332 0.1415 29.3434 29.177 2.8447 2.8284 0.3092 0.3057</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>0.2002 0.2121 41.1484 40.7889 4.0409 3.9747 0.4435 0.4338</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>0.322 0.3404 48.6608 48.5744 4.801 4.7639 0.5282 0.5224</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Random Initialization. Random initial task clusters for each robot were generated as illustrated in Fig. 2. The algorithms cSCA and gcSCA are evaluated for 3 cooling schedule ratios $\beta$, representing slow ($\beta=1.001$), medium ($\beta=1.01$), and fast ($\beta=1.1$) algorithm convergence. It is seen from Fig. 3(a) that cSCA and gcSCA with random initialization improve or match the performance of the greedy algorithms since the MCI is always nonnegative. Of course, this is at the expense of additional ACs as seen in Fig. 3(b), which leads to longer CPU times as seen in Table I. However, Figs. 3(a) and 3(b) with Table I also show that
the parameter $\beta$ enables cSCA and gcSCA to trade off optimality and computational/communication performance. For example, as $\beta$ increases from 1.001 to 1.1, the annealing process cools down more rapidly so that ACs and computational times are reduced for both cSCA and gcSCA. In particular, for cSCA the AC interval changes from $[8,4000]$ to $[6,45]$ with a maximum decrease of 98.89% and the computation time interval changes from $[1.0032 \, s, 48.6608 \, s]$ to $[0.0123 \, s, 0.5282 \, s]$ with a maximum decrease of 98.91%. This is accomplished at the expense of MCI, which changes from $[4.1\%, 16.7\%]$ to $[0.1\%, 6\%]$ with a maximum decrease of 95.75%. It is interesting to note from Fig. 3(a) that for the smallest cooling schedule ratio, $\beta = 1.001$, the MCI of cSCA is in the interval $[4.1\%, 16.7\%]$ and is always greater than the MCI for gcSCA, which is in the interval $[2.3\%, 14.5\%]$. In contrast, for the larger cooling schedule ratio, $\beta = 1.01$, the MCI of gcSCA is in the interval $[0.16\%, 13.6\%]$ and is always greater than the MCI for cSCA, which is in the interval $[0.12\%, 13.1\%]$ with a maximum increase of 8.43%. Fig. 3 and Table I also show that for medium to fast annealing the ACs and computational times are greater for cSCA than gcSCA, even when the MCI of cSCA is less than that of gcSCA. For example, for $\beta = 1.01$ the ACs and computational times of gcSCA, which lie in the respective intervals $[3,69]$ and $[0.0248 \, s, 4.7639 \, s]$, are always lower than the ACs for cSCA, which lie in the intervals $[7,412]$ and $[0.1098 \, s, 4.801 \, s]$ with maximum respective decreases of 83.25% and 77.41%.

As discussed more quantitatively above, Fig. 3 shows that for medium to fast annealing, e.g., $\beta = 1.01$ or $\beta = 1.1$, gcSCA tends to converge faster than cSCA in terms of ACs by an order of magnitude, while actually exceeding cSCA in MCI. The reason for this is that the uphill random walk that is a part of cSCA (Step 8 in Algorithm 1) is inefficient when the annealing is sufficiently fast. The purpose of the uphill random walk is to enable the optimization to escape local minima. However, when the annealing is fast, the optimization usually converges to a local minimum and the uphill random walk simply makes the optimization less efficient. Hence, if medium to fast annealing is needed (e.g., during a mission), it may be advisable to turn off the uphill random walk. If slower annealing is allowed (e.g., at the beginning of a mission), the uphill movement should be used to increase performance.

### 3.1.2 Initialization with a Greedy Solution

The goal of this study was to determine whether it is beneficial to initialize SCA with a greedy solution as opposed to a random initialization as was done previously. Only the results corresponding to the cooling schedule ratio $\beta = 1.01$ are presented for simplicity and clarity, but the trends are also true for other choices of cooling schedule ratios, e.g. $\beta = 1.001$ and $\beta = 1.1$. Fig. 4 compares the average MCIs and ACs when cSCA and gcSCA were initialized with the allocations corresponding to the lowest costs achieved by the set of greedy auctions $\{SA, LBSA, PA, LBPA\}$ and with random initialization. From Fig. 4(a) it is seen that the average MCIs for both cSCA and gcSCA resulting from this greedy initialization lie in the respective intervals $[2.14\%, 15.02\%]$ and $[3.09\%, 15.42\%]$, and are always greater than the average MCIs of both cSCA and gcSCA with random initializations, which lie in the intervals $[0.12\%, 13.1\%]$ and $[0.0248 \, s, 4.7639 \, s]$, respectively.
3.2 Comparison of cSCA, gcSCA, and the Greedy Auction Algorithms with Varying Number of Robots

Fig. 5. Mean cost improvement vs. the number of robots and the number of tasks for cSCA and gcSCA for $\beta = 1.01$
The purpose of adding the number of robots as a variable in the random simulations is twofold. First, it is desired to provide evidence that the relationship between the MCIs and ACs of cSCA and gcSCA observed in Section 3.1 for 3 robots extends to an arbitrary number of robots. Second, it is intended to generate curves that can be used to determine the number of robots needed for a mission that is specified by some number (or range of numbers) of possible tasks in a specified region. Hence, in what follows 1000 random scenarios are again studied for a given number of robots and tasks with the number of robots ranging from 2 to 10 and the number of tasks ranging from 10 to 260 in increments of 10.

The simple answer to the first issue is that the trends observed in Section 3.1 were evidenced in the simulations involving a varying number of robots for the three values of \( \beta \) considered previously. That is, cSCA incurs lower costs than gcSCA for slow annealing (e.g., \( \beta = 1.001 \)), gcSCA incurs lower costs than cSCA for faster annealing (e.g., \( \beta = 1.01 \) and \( \beta = 1.1 \)), and gcSCA provides substantial (sometimes one order of magnitude) improvements in the ACs. This is illustrated for \( \beta = 1.01 \) by Fig. 5 and Fig. 6, which for both cSCA and gcSCA display the MCIs (Fig. 5) and ACs (Fig. 6). In these figures the solid lines denote the interpolated real data and the surfaces are obtained by interpolating the solid lines. The trends for the range of robots considered remain the same as those for 3 robots (see Fig. 3). It is interesting to note that in Fig. 6(b) the number of ACs in gcSCA decreases with the number of robots after the number of robots exceeds a certain threshold (in this case 6). The reason for this phenomenon is not clear. However, it appears to somehow be caused by the combined effect of using the (greedy) constrained Prim's algorithm (see Step 2 of Section 2.2.1) in conjunction with the greedy Gibb's sampler.

Fig. 7 displays the costs (in this case for gcSCA with \( \beta = 1.01 \)) as a function of the number of robots and tasks. It shows that as the number of tasks increases, a substantial performance improvement (i.e., time savings) can be achieved by adding a small number of robots. For example in Fig. 7(b), which shows the costs for 260 tasks, the cost corresponding to 2 robots is 452.5 hrs, while the costs with 4 robots improves to 149.5 hrs. In general for a fixed number of tasks, the corresponding.
Fig. 7. Mean cost vs. the number of robots and the number of tasks, and its “slice” in a frontal plane for gcSCA with $\beta = 1.01$.

“slice” of a 3-D curve such as Fig. 7(b) may be used to trade off performance vs. the number of robots and hence provides a guideline for choosing the desired number of robots for the expected mission.

4. EXPERIMENTAL RESULTS FOR DISTRIBUTED IMPLEMENTATION OF SCA

As previously discussed, distributed auctions are needed due to limited communication between robots. This section uses numerical experiments to evaluate the efficacy of the distributed SCA (dSCA) approach described in Section 2.3. As in Section 3, random scenarios were simulated in a $10000 \times 10000m$ area and the speeds for each of the robots were assumed to be constant and were chosen randomly from the interval $(0 \, m/s, 20 \, m/s]$ assuming a uniform distribution. Considering a given robot, the completion time for a given task was chosen as in the first paragraph of Section 3, and again, the cost function is a MinMax cost function in (7) corresponding to the mission completion time. The SCA parameters used were as before: initial temperature, $T = 1000$; termination temperature, $T_{cut} = 20$; and the cooling schedule ratio, $\beta = 1.01$.

Distributed SCA is used during a mission and hence the number of ACs associated with each tournament should be limited (say $< 10^2$). As a result, in this section dSCA utilizes the gcSCA algorithm with $\beta = 1.01$ since the results of Section 3 show that good performance is achieved in this case with a reasonable number of ACs.

As will be discussed in detail in Section 4.1, four fundamental network topologies are used to define four benchmark auction communication scenarios. The simulation results are used to determine how closely the cost of distributed auctioning can approach the cost of centralized auctioning and to compare the corresponding convergence properties. Section 4.2 introduces two metrics used in the evaluation of dSCA. The simulations described in Section 4.3 are based on static scenarios in which the tasks are fixed, while the simulations of Section 4.4 are based on dynamic scenarios in which new tasks are introduced after the auctioning commences. Both types of scenarios can occur in practice.
Fig. 8. Four benchmark auction configuration and auction patterns for evaluating distributed auctioning with $k$ robots.

4.1 Four Benchmark Auction Configuration and Auction Patterns

The four benchmark auction communication scenarios, derived from fundamental network topologies [Groth and Skandier 2005], are shown in Fig. 8. Fig. 8(a) represents scenarios in which the robots are widely distributed and there is minimal communication between them. Fig. 8(b) demonstrates scenarios in which each robot can correspond with two additional robots, which is slightly less restrictive communication than that of Fig. 8(a). Fig. 8(c) corresponds to scenarios in which several robots may be in communication range of a given robot, but the communication between neighboring robots is intentionally or unintentionally interrupted. Fig. 8(d) represents scenarios in which one robot has communication channels with several other robots, but in general the communication between robots is restricted. A fully connected network topology is not shown, but it would correspond to centralized auctioning. It is contended that if dSCA works well in these diverse auction communication scenarios, it is strong evidence that it will work well in real world scenarios.

For the row and hybrid benchmarks corresponding respectively to Figs. 8(a) and 8(d) the initial robot positions were chosen to be evenly distributed along the one edge of the scenario area. For the circular and mesh benchmarks corresponding respectively to Figs. 8(b) and 8(c) the origins of the circles defining the positions of the robots were at the center of the 10000 m × 10000 m area and the diameters of the circles were chosen to be 5000 m. For each of the benchmarks dSCA was initialized using the set of fast greedy algorithms \{SA,LBSA,PA,LBPA\} (see Section 3.1.2).

4.2 Two Metrics for Evaluation of dSCA

The efficacy of dSCA is measured by comparing the resultant global cost with the corresponding gcSCA global cost. This leads to the following definition for optimization efficiency.

**Definition 3** The optimization efficiency for scenario \( r \) is denoted by \( \eta_r \in (0, 1] \) and defined by

\[
\eta_r \triangleq \frac{C^*_r}{C_r},
\]

where \( C^*_r \) is the global cost resulting from the application of gcHYSCA and \( C_r \) is the global cost resulting from the application of dHYSCA.

A tournament corresponds to one round of distributed auctioning in which one of the robots serves as the auctioneer and leads an auction with the robots that are within communication range. To quantify the extent of robot interaction in the tournaments of the distributed auctioning the concept of tournament participation index is introduced in the following definition. Increasing values of this index corresponds to increasing communication between the robots.

**Definition 4** The Tournament Participation Index (TPI)\(^5\) for \( k \) robots is
denoted by $\zeta(k) \in (0, 1]$ and defined by

$$\zeta(k) \triangleq \frac{\sum_{p=1}^{k} b^2(p)}{k^2} = \frac{\sum_{p=1}^{k} b^2(p)}{k^3}, \quad (15)$$

where $b(p)$ is the number of robots that participate in the regional auction in which robot $h_p$ is the auctioneer. Hence $\zeta(k)$ is the mean of $b^2(p)$ for the $k$ robots, normalized so that it lies in the interval $(0, 1]$.

To better understand the TPI, especially the normalization, note that full communication among all the robots results in $b(p) = k$ for $p = 1, 2, \ldots, k$. Hence, in this case (15) yields

$$\zeta(k) = \frac{\sum_{p=1}^{k} k}{k^3} = \frac{k^2}{k^3} = 1. \quad (16)$$

Hence, the TPI $\zeta(k) = 1$ corresponds to full communication between each of the robots.

Also, note that for the benchmark in Fig. 8(a)

$$b(i) = \begin{cases} 2, & i = 1, k \text{ (i.e., for the end robots)} \\ 3, & \text{otherwise (i.e., for the middle robots)}, \end{cases} \quad (17)$$

while for the benchmark in Fig. 8(b)

$$b(i) = 3, \quad \text{for all } i \quad (18)$$

Hence, if we let $\zeta_{row}(k)$, $\zeta_{cir}(k)$, $\zeta_{mesh}(k)$, and $\zeta_{hyb}(k)$ denote, respectively, the TPIs corresponding to Figs. 8(a), 8(b), 8(c) and 8(d), then the TPIs corresponding to the simulations presented below are given in Table II.

**Table II. Tournament Participation Indices corresponding to the four benchmark auction communication scenarios of Fig. 8**

<table>
<thead>
<tr>
<th>no. of robots ($k$)</th>
<th>$\zeta_{row}(k)$</th>
<th>$\zeta_{cir}(k)$</th>
<th>$\zeta_{mesh}(k)$</th>
<th>$\zeta_{hyb}(k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.41</td>
<td>0.56</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>0.31</td>
<td>-</td>
<td>0.68</td>
<td>0.35</td>
</tr>
</tbody>
</table>

**4.3 Static Scenarios**

In the static scenarios a fixed number of tasks are given at the outset of auctioning and subsequently no tasks are changed or added. For a given number of tasks and auction communication scenario each simulation involved 1000 random scenarios. The results for the four benchmarks of Fig. 8 with the number of robots $k$ equal to either 4 or 6 (see Table II) are shown in Fig. 9. In this figure the solid lines denote the interpolated real data and the surfaces are obtained by interpolating the solid lines. As predicted by Proposition 1, the cost efficiency increases monotonically

---

5TPI is similar to but different than the degree or connectivity in networks or graph theory since there are no redundant connections between two robots and $b(p)$ counts the robots instead of the links.

as the number of tournaments increases. Also, it is seen in Fig. 9 that as the TPI increases from 0.31 to 0.35 to 0.41 to 0.68, the mean cost efficiencies during the tournaments progressively increase (from the interval [0.65,0.76] to the interval [0.77,0.9] to the interval [0.82,0.94] to the interval [0.88,1]). Therefore, TPI serves as a good predictor of the relative optimization efficiency for dSCA in static scenarios.

\[
\begin{align*}
\text{(a) 4 robots in row benchmark: } & \zeta_{\text{row}}(4) = 0.41 \\
\text{(b) 6 robots in row benchmark: } & \zeta_{\text{row}}(6) = 0.31 \\
\text{(c) 6 robots in mesh benchmark: } & \zeta_{\text{mesh}}(6) = 0.68 \\
\text{(d) 6 robots in hybrid benchmark: } & \zeta_{\text{hyb}}(6) = 0.35
\end{align*}
\]

Fig. 9. The mean optimization efficiency vs. tournaments and number of tasks for two cases of the row benchmark of Fig. 8(a), one case of the mesh benchmark of Fig. 8(c), and one case of the hybrid benchmark of Fig. 8(d)

An important question is how decentralized SCA (dSCA) compares in terms of on-line implementation speed with centralized SCA (specifically, gcSCA). The actual speeds are dependent upon the processor used to implement the SCA algorithms. In addition, without real-time implementation it is difficult to gauge the time that it takes to communicate between the auctioneer and robots. Hence, in order to use simulations to provide a meaningful and fair comparison between algorithms, we assume that the speed of the algorithms is primarily dependent upon the number of cost calculations computed by the non-auctioneer robots, since the computation of these costs and the communication of these costs to the auctioneer
should be the primary determinants of the on-line implementation speed of the algorithms. These calculations take into account the parallel computations associated with (11) and (12) (see the discussion just after these equations in Section 2.2.1). That is, when a cost is calculated in parallel with another cost, only one cost calculation is counted. Fig. 10 uses this cost calculation metric to compare the on-line implementation speed of gcSCA and dSCA. For Tournaments 3, 6 and 9 on average dSCA takes respectively 258.4%, 621.23%, and 989.87% longer than gcSCA. This is indicative of the increased computational requirements associated with the improvement in robustness of distributed algorithms.

(a) Results for the row benchmark of Fig. 8(a) with new tasks introduced between Tournaments 4 and 5: $\zeta_{row}(4) = 0.41$

(b) Results for the circle benchmark of Fig. 8(b) with new tasks introduced between Tournaments 3 and 4: $\zeta_{cir}(4) = 0.56$

Fig. 11. Mean optimization efficiency vs. number of tournaments and the ratio of new tasks to initial tasks: 4 robots, 120 initial tasks

4.4 Dynamic Scenarios
In the dynamic scenarios a specific number of tasks are given at the outset of auctioning and after a certain number of tournaments new tasks are added. Sub-
(a) Results for the row benchmark of Fig. 8(a): $\zeta_{\text{row}}(6) = 0.31$

(b) Results for the mesh benchmark of Fig. 8(c): $\zeta_{\text{mesh}}(6) = 0.68$

(c) Results for the hybrid benchmark of Fig. 8(d): $\zeta_{\text{hybrid}}(6) = 0.35$

Fig. 12. Mean optimization efficiency vs. number of tournaments and the ratio of new tasks to initial tasks: 6 robots, 120 initial tasks with new tasks introduced between Tournaments 7 and 8

sequently, the tournaments continue and assign the new tasks to the appropriate robots. The number of new tasks introduced varies with each simulation and is some ratio times the number of the initial tasks. For a given number of initial tasks and auction communication scenario each simulation involved 1000 random scenarios. The results are shown in Fig. 11 and Fig. 12. As in Fig. 9, in these figures the solid lines denote the interpolated real data and the surfaces are obtained by interpolating the solid lines. Note that Fig. 11(a) shows that the optimization efficiency may actually decrease after new tasks are introduced, but both Fig. 11(a) and Fig. 11(b) show that the distributed auctioning can accommodate the new tasks and increase the optimization efficiency as the number of tournaments increases. Also, note that due to a greater TPI, the mean cost efficiency is greater in Fig. 11(b) than in Fig. 11(a). Similar phenomena can be observed in Fig. 12. Therefore, as for the static scenarios, TPI serves as a good predictor of the relative optimization efficiency for dSCA in dynamic scenarios.
5. CONCLUSIONS

Stochastic Clustering Auction (SCA) is the first stochastic auction framework using global optimization. Specifically, it is based on stochastically moving and swapping tasks between the clusters assigned to each robot. The fundamental SCA algorithm has the ability to consider both downhill and uphill movements, the latter providing the ability of SCA to avoid local minima. The underlying optimization algorithm is simulated annealing, which has a key parameter $\beta$, the cooling schedule ratio, that can be used to determine the convergence time of SCA. The ability to choose $\beta$ and turn the uphill movements on and off provides SCA with the ability to trade off the converged algorithm cost with computational and communication efficiency, a novel feature of SCA.

Using extensive simulations involving random scenarios, SCA was evaluated for centralized auctioning. Special attention was given to comparing the performance of SCA with two standard greedy auction algorithms, parallel and sequential auctioning, and two closely related greedy algorithms, which were defined in Appendix B. In addition, centralized SCA with uphill movements, denoted cSCA, and centralized SCA without uphill movements, denoted greedy cSCA or gcSCA, were compared. The results showed that it is beneficial in terms of performance and algorithm convergence to initialize both cSCA and gcSCA with an allocation obtained from a greedy algorithm as opposed to initializing them with a random allocation. Perhaps even more interesting, it was seen that if $\beta$ is chosen sufficiently large (e.g., $\beta = 1.01$ or $\beta = 1.1$) so that the annealing is sufficiently fast, gcSCA actually outperforms cSCA in terms of both performance and convergence. This is because when the annealing is fast, both cSCA and gcSCA converge to a local minimum, and the uphill movements in cSCA simply introduce inefficiency. Ultimately, it seems that at the beginning of a mission, when time and communication permit, it may be advisable to use cSCA with slow annealing, but during a mission gcSCA should be preferred.

The random simulation for cSCA also resulted in 3-D curves that show cost vs. number of robots and number of tasks. Given a fixed number of tasks, a 2-D slice of the 3-D curve shows the tradeoff between cost and number of robots. This enables an appropriate choice of the number of robots for a given mission that has an expected number of tasks. The same simulation approach can be used with any auction algorithm to produce a similar 3-D surface.

Distributed SCA, denoted as dSCA, was based on applying gcSCA regionally and enabling each robot to serve as the auctioneer in a fixed rotation pattern. The evaluation of dSCA on random scenarios relied on the use of four benchmark auction communication scenarios derived from four fundamental network topologies and considered both static and dynamic scenarios. The benchmarks were chosen to represent the type of robot communication that may be experienced in real-world scenarios, e.g., communication between widely distributed robots or tight communication between a subset of the robots. The simulation results uniformly showed that in terms of mean cost dSCA continuously improved the global performance each time one of the robots completed its tournament (i.e., its auction process based on gcSCA). The amount of robot communication was measured by the Tournament Participation Index (TPI), with increasing TPI denoting increasing average
robot communication. It was seen that the ability of dSCA to approach the cost achieved by gcSCA is dependent on the TPI. However, dSCA was always far more computationally intensive than gcSCA. This appears to be the fundamental sacrifice associated with achieving robustness with respect to limited communication between robots.

It should be recognized that the SCA algorithm presented here is only the first stochastic clustering algorithm based on global optimization. It is envisioned that additional algorithms will be developed that yield better performance with reduced computational and communication requirements based on more complex movements between clusters. A possible basis for one such algorithm is the Swendsen-Wang methodology [Barbu and Zhu 2005; Swendsen and Wang 1987]. SCA algorithms can be extended to handle the variety of scenarios that can occur in real world practice such as: 1) tasks that have interdependence [Zlot and Stentz 2006], 2) tasks for which the completion duration time will change over time as robots discover more information about their environment, and 3) robot malfunction and communication delay [Lumezanu et al. 2008; Puschini et al. 2009]. Another consideration is the development of distributed SCA algorithms that use multiple auctioneers working in parallel for various topologies of the auction pattern, such as random sequencing in the 4 benchmark networks, the Watts-Strogatz small-world model or the Barabási-Albert scale-free network [Albert and Barabási 2002]. This will require the development and incorporation of mechanisms for conflict resolution. SCA will also be experimentally implemented using real robots.

ACKNOWLEDGMENTS
This work was sponsored, in part, by the US Army Research Laboratory under Cooperative Agreement W911NF-06-2-0041, by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD 19-01-2-0012, and by the Army Research Office under grant W911NF-09-1-0568. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ARL or the US Government.

The authors thank Xiuwen Liu, Oscar Chuy, Jr. and Adrian Barbu for helpful discussions.

REFERENCES


A. USE OF SCA FOR DISTRIBUTED TASK ALLOCATION

This section first presents the concept of the regional cost. Then, a simple, but rigorous proposition is presented to motivate the use of SCA for distributed auctioning.

A.1 Regional Cost

Assume that robot $h_q$ is the auctioneer for the set of heterogenous robots $H_L$ defined by

$$H_L = \{h_p : p \in L\},$$

(19)

where

$$q \in L \subseteq \{1, 2, \ldots, k\}.$$  

(20)

It follows from (20) that

$$H_L \subseteq H,$$

(21)

and hence $h_i$ is the auctioneer for a distributed auction. If the initial global allocation is given by

$$A^{(0)} = \{a_1^{(0)}, a_2^{(0)}, \ldots, a_k^{(0)}\},$$

(22)

then the set of initial tasks in the distributed auction is

$$T_D^{(0)} = \{t \in a_p^{(0)} : p \in L\} \subseteq T.$$  

(23)

An allocation associated with the distributed auction is given by

$$A_L = \{a_p : p \in L\},$$

(24)
where the set of tasks in the allocation $A_L$ is $T_D$, i.e.,

$$T_D = \{ t \in a_p : p \in L \}. \quad (25)$$

**Definition 5** The regional cost associated with the robots $H_L$ and the tasks $T_D$ is defined by

$$C(A_L) \triangleq \sum_{p \in L} c_p(a_p) \quad (26)$$

when the global cost $C(A)$ is given by (6), and

$$C(A_L) \triangleq \max_{p \in L} c_p(a_p) \quad (27)$$

when the global cost $C(A)$ is defined by (7).

Now, let $L^c$ denote the complement of $L$, i.e.,

$$L^c \cup L = \{ 1, 2, \ldots, k \}. \quad (28)$$

Then, in terms of the regional cost, the global cost is given by

$$C(A) = C(A_L) + \sum_{p \in L^c} c_p(a_p) \quad (29)$$

or

$$C(A) = \max \left( C(A_L), \max_{p \in L^c} c_p(a_p) \right). \quad (30)$$

The expressions in (29) and (30) are important in the proof of the theory in Appendix A.2.

A.2 Theory for Distributed Stochastic Clustering Auction

**Proposition 1** Given an initial global allocation $A^{(0)}$ defined by (22) and a region $L$, let $A_L^{(0)} = \{ a_p^{(0)} : p \in L \}$. Suppose $A_L^*$ is the regional allocation resulting from a regional auction, and assume

$$C(A_L^*) \leq C(A_L^{(0)}). \quad (31)$$

If $A'$ denotes the global allocation in which the robots $H_L$ have the allocation $A_L^*$ and the remaining robots, i.e., $\{ h_p : p \in L^c \}$ have the allocation $\{ a_p^{(0)} : p \in L^c \}$, then

$$C(A') \leq C(A^{(0)}), \quad (32)$$

i.e., the new global cost will be at least as small as the global cost of the initial global allocation.

**Proof:** The complete proof is omitted. It relies on the decompositions in (29) and (30). Intuitively speaking, if the global cost can be decomposed into the sum or maximum of two costs, then reducing one of these costs while holding the remaining cost fixed will cause the global cost to reduce or remain the same. □

Since cSCA and gcSCA always reduce or hold the cost constant (see Step (6) of Section 2.2.1), motivated by Proposition 1, distributed SCA simply applies cSCA or gcSCA regionally. Each robot becomes the auctioneer in some rotation pattern.
B. DESCRIPTION OF FOUR GREEDY AUCTION ALGORITHMS

Before comparing SCA with other auctioning methods, four greedy auction algorithms are introduced in this section: the Sequential (single-item) Auction (SA) and the Parallel Auction (PA), which are standard auction methods in the existing literature [Dias et al. 2006; Gerkey and Mataric 2002; Koenig et al. 2007; Simmons et al. 2007; Zlot and Stentz 2006], and their variants, the Look-Back Sequential (single-item) Auction (LBSA) and the Look-Back Parallel Auction (LBPA), given in this section along with descriptions of SA and PA. LBSA and LBPA sometimes yield better performance than their better-known respective counterparts, SA and PA, while having similar computational requirements.

Algorithm 2 principal mechanisms for SA and LBSA

\[
\begin{aligned}
1 & : \text{for each robot } h \in \mathcal{H} \text{ do} \\
2 & : \quad T(h) \leftarrow \emptyset \\
3 & : \text{end for} \\
4 & : \text{repeat} \\
5 & : \quad \text{for each robot } h \in \mathcal{H} \text{ do} \\
6 & : \quad \text{for each task } t \in \mathcal{T} \text{ do} \\
7 & : \quad \quad \text{if } T(h) \text{ is empty then} \\
8 & : \quad \quad \quad \text{bid}(h, t) \leftarrow SC(h, T(h) \cup t) - SC(h, T(h)) \\
9 & : \quad \quad \quad \text{else} \\
10 & : \quad \quad \quad \text{bid}(h, t) \leftarrow SC(h, T(h) \cup t) - SC(h, T(h)) + \text{bid}(h, t') \{\text{the addition of bid}(h, t') \text{ only for LBSA}\} \\
11 & : \quad \quad \text{end if} \\
12 & : \quad \text{submit}(h, t, \text{bid}(h, t)) \\
13 & : \text{end for} \\
14 & : \text{end for} \\
15 & : \text{the auctioneer allocates the task } t(h_{\text{min}}) \text{ to winner robot } h_{\text{min}} \\
16 & : \mathcal{T}_{\text{un}} \leftarrow \mathcal{T}_{\text{un}} \setminus t(h_{\text{min}}) \\
17 & : \mathcal{T}(h) \leftarrow \mathcal{T}(h) \cup t(h_{\text{min}}) \\
18 & : \text{until } \mathcal{T}_{\text{un}} = \emptyset
\end{aligned}
\]

Algorithm 2 describes SA and LBSA, while Algorithm 3 describes PA and LBPA. In these algorithms \(\mathcal{H}\) denotes a set of heterogeneous robots and \(\mathcal{T}\) denotes a set of tasks. \(\mathcal{T}_{\text{un}}\) is the set of unallocated tasks in \(\mathcal{T}\) and initialized as \(\mathcal{T}\). \(T(h)\) is the set of tasks owned by robot \(h \in \mathcal{H}\) (initially the empty set). \(SC(h, \mathcal{T})\) is the smallest cost for completing all tasks in \(\mathcal{T}\) from the current location of robot \(h\). A bidding table is constructed based on the current bid, \(\text{bid}(h, t) = SC(h, \mathcal{T}(h) \cup t) - SC(h, \mathcal{T}(h))\) [Koenig et al. 2006]. The bid \(\text{bid}(h, t')\) denotes the last bid won for robot \(h_{\text{min}}\) and \(t(h_{\text{min}})\) the task won by robot \(h_{\text{min}}\). The winner robot and task combination is defined by (33).

\[
(h_{\text{min}}, t(h_{\text{min}})) = \arg \min_{h \in \mathcal{H}, t \in \mathcal{T}} \text{bid}(h, t) \quad (33)
\]

For LBSA Step 10 of Algorithm 2 and for LBPA Step 10 of Algorithm 3 take into account the previous bids when considering the cost of the current bid. These steps differentiate these auction algorithms from SA and PA.
Previous studies [Zhang et al. 2008; Barbu and Zhu 2005] reveal the performance and algorithm convergence benefits of initializing an SCA with an allocation obtained from a greedy algorithm as opposed to initializing them with a random allocation. Thus, the lowest cost allocation from the set of greedy auctions \{SA,LBSA,PA,LBPA\} is used to initialize SCAs.

**Algorithm 3** principal mechanisms for PA and LBPA

```
1: for each robot \( h \in \mathcal{H} \) do
2:  \( \mathcal{T}(h) \leftarrow \emptyset \)
3: end for
4: repeat
5:  for each robot \( h \in \mathcal{H} \) do
6:    for each task \( t \in \mathcal{T} \) do
7:      if \( \mathcal{T}(h) \) is empty then
8:        bid\((h,t)\) \( \leftarrow SC(h,\mathcal{T}(h) \cup t) - SC(h,\mathcal{T}(h)) \)
9:      else
10:        bid\((h,t)\) \( \leftarrow SC(h,\mathcal{T}(h) \cup t) - SC(h,\mathcal{T}(h)) + \text{bid}(h,t') \) \{the addition of bid\((h,t')\) only for LBPA\}
11:      end if
12:    submit\((h,t,\text{bid}(h,t))\)
13:  end for
14:  the auctioneer allocates the task \( t(h_{\text{min}}) \) for winner robot \( h_{\text{min}} \) (modulo tie breaking)
15:  \( \mathcal{T}_{\text{un}} \leftarrow \mathcal{T}_{\text{un}} \setminus t(h_{\text{min}}) \)
16:  \( \mathcal{T}(h) \leftarrow \mathcal{T}(h) \cup t(h_{\text{min}}) \)
17: end for
18: until \( \mathcal{T}_{\text{un}} = \emptyset \)
```