Consensus-Based Decentralized Auctions for Robust Task Allocation

Han-Lim Choi, *Member, IEEE*, Luc Brunet, and Jonathan P. How, *Senior Member, IEEE*

***Abstract*—This paper addresses task allocation to coordinate a fleet of autonomous vehicles by presenting two decentralized algorithms: the consensus-based auction algorithm (CBAA) and its generalization to the multi-assignment problem, i.e., the consensus-based bundle algorithm (CBBA). These algorithms utilize a market-based decision strategy as the mechanism for decentralized task selection and use a consensus routine based on local communication as the conflict resolution mechanism to achieve agreement on the winning bid values. Under reasonable assumptions on the scoring scheme, both of the proposed algorithms are proven to guarantee convergence to a conflict-free assignment, and it is shown that the converged solutions exhibit provable worst-case performance. It is also demonstrated that CBAA and CBBA produce conflict-free feasible solutions that are robust to both inconsistencies in the situational awareness across the fleet and variations in the communication network topology. Numerical experiments confirm superior convergence properties and performance when compared with existing auction-based task-allocation algorithms.**

***Index Terms*—Distributed robot systems, networked robots, task allocation for multiple mobile robots.**

I. Introduction

COOPERATION among a fleet of robotic agents is necessary in order to improve the overall performance of any mission. Many different methods exist that enable a group of such agents the ability to distribute tasks among themselves from a known task list. Centralized planners [2]–[8] communicate their situational awareness (SA) to a centralized server that generates a plan for the entire fleet. These types of systems are useful since they place much of the heavy processing requirements safely on the ground, thus making the robots smaller and cheaper to build. On the other hand, agents must consistently communicate with a fixed location, thus reducing the possible mission ranges that the fleet can handle, as well as creating a single point of failure in the mission.

Manuscript received December 7, 2008; revised April 14, 2009. First published June 12, 2009; current version published July 31, 2009. This paper was recommended for publication by Associate Editor T. Murphey and Editor W. K. Chung upon evaluation of the reviewers' comments. This work was supported in part by the Air Force Office of Scientific Research (AFOSR) Small Business Technology Transfer under Contract FA9550-06-C-0088 and in part by the AFOSR under Grant FA9550-08-1-0086. This paper was presented in part at the American Institute of Aeronautics and Astronautics Guidance, Navigation, and Control Conference, Honolulu, HI, 2008, while refined theoretical proofs and extensive numerical results have been included in this paper.

H.-L. Choi and J. P. How are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: hanlimc@mit.edu; jhow@mit.edu).

L. Brunet is with Frontline Robotics, Ottawa, ON K4P 1A2, Canada (e-mail: lbrunet@frontline-robotics.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2009.2022423

Some types of decentralized methods have thus been developed by instantiating the centralized planner on each agent in order to increase the mission range, as well as remove the single point of failure [9]–[12]. These methods often assume perfect communication links with infinite bandwidth since each agent must have the same SA. If this is not the case, it has been shown that realistic networks with limited communication can significantly affect the fleet's ability to coordinate their actions [13]. In this case, inconsistencies in the SA might cause conflicting assignments, since each agent will be performing the centralized optimization with a different information set. Thus, decentralized algorithms generally make use of consensus algorithms [14]–[19] to converge on a consistent SA before performing the assignment [20]. These consensus algorithms can guarantee convergence of the SA over many different dynamic network topologies [21]–[23], thus allowing the fleet to perform the assignment in highly dynamic and uncertain environments.

Although consensus algorithms allow a fleet of vehicles to converge on the SA and perform an assignment over many generic network topologies, convergence to a consistent SA may take a significant amount of time and can often require transmitting large amounts of data to do so [24]. This can cause severe latency in low-bandwidth environments and can substantially increase the time it takes to find an assignment for the fleet. To resolve this, approaches that do not aim for perfect consensus on the SA have been suggested: In [24], robustness to inconsistent SA was enhanced by allowing agents to communicate plans as well as SA, while in [25], communication occurrence was restricted only to the cases where there is mismatch between plans based on the local knowledge and the estimated global knowledge. However, these algorithms might still take a significant amount of time to produce a final solution, because the first requires each agent to receive plans from all other agents, and the second might still need perfect consensus to guarantee conflict-free solutions.

Auction algorithms [26]–[29] are another method for task assignment that have been shown to efficiently produce suboptimal solutions [30]. Generally, agents place bids on tasks, and the highest bid wins the assignment. The traditional way of computing the winner is to have a central system acting as the auctioneer to receive and evaluate each bid in the fleet [31]–[33]. Once all of the bids have been collected, a winner is selected based on a predefined scoring metric. In other formulations, the central system is removed, and one of the bidders acts as the auctioneer [34]–[38]. In these types of algorithms, agents bid on tasks with values based solely on their own SA. It is known that each task will only be assigned to a single agent since only one agent is selected by the auctioneer as the winner. Because

of this, most auction algorithms can naturally converge to a conflict-free solution even with inconsistencies in their SA. The downside of these approaches is that the bids from each agent must somehow be transmitted to the auctioneer. This limits the network topologies that can be used since a connected network is required between the agents in order to route all of the bid information. A common method to avoid this is to sacrifice mission performance by running the auction solely within the set of direct neighbors of the auctioneer [39], [40].

Thus, algorithms that use consensus before planning are generally more robust to network topologies, while traditional auction approaches are computationally efficient and robust to inconsistencies in the SA. This work aims at combining both approaches in order to take advantage of properties from both allocation strategies. This paper employs the auction approach for decentralized task selection and the consensus procedure for decentralized conflict resolution. The key difference from previous consensus-based methods is that the consensus routine is used to achieve agreement on the winning bid values instead of SA. For single-assignment problem in which at most one task can be assigned to a single agent, the consensus-based auction algorithm (CBAA) is presented; then, this algorithm is extended to the multi-assignment problem in which a sequence of multiple tasks is assigned to each agent by developing the consensus-based bundle algorithm (CBBA).

Various efforts have been made in the literature to extend the auction class of algorithms to the multi-assignment case. In many cases, this is done by running sequential auctions and awarding a single task at a time until there are no remaining tasks left to assign [34], [40], [41]. Bundle approaches [42]–[45] have been developed that group common tasks into bundles and allowing agents to bid on groups rather than the individual tasks. By grouping similar tasks, these types of algorithms will converge faster than their sequential counterparts and may have improved value in the assignment since they can logically group tasks that have commonalities. However, difficulties can arise in the computational cost for enumerating all possible bundle combinations and determining the winner among these bundles. The winner determination has been shown to be NP-complete [46], and only heuristic methods [47]–[49] are available. CBBA, however, builds a single bundle and bids on the included tasks based on the improvement they provide to the bundle. Computation is reduced by considering only a single bundle while convergence times are improved over sequential auctions since multiple tasks can be assigned in parallel. In this paper, it is analytically shown that CBBA produces the same solution as some centralized sequential greedy procedures, and this solution guarantees 50% optimality. Also, numerical simulations verify that the proposed algorithm outperforms existing sequential auction methods in terms of quick convergence and small optimality gap.

II. Background

A. Task-Allocation Problems

The goal of task allocation is, given a list of N_t tasks and N_u agents, to find a conflict-free matching of tasks to agents that maximizes some global reward. An assignment is said to be free of conflicts if each task is assigned to no more than one agent. Each agent can be assigned a maximum of L_t tasks, and the assignment is said to be completed once $N_{\min} \triangleq \min\{N_t, N_u L_t\}$ tasks have been assigned. The global objective function is assumed to be a sum of local reward values, while each local reward is determined as a function of the tasks assigned to each agent. The task-assignment problem described before can be written as the following integer (possibly nonlinear) program with binary decision variables x_{ij} that indicate whether or not task j is assigned to agent i

$$\max \quad \sum_{i=1}^{N_u} \left(\sum_{j=1}^{N_t} c_{ij}(\mathbf{x}_i, \mathbf{p}_i) x_{ij} \right)$$

subject to

$$\begin{aligned} \sum_{j=1}^{N_t} x_{ij} &\leq L_t \qquad \forall i \in \mathcal{I} \\ \sum_{i=1}^{N_u} x_{ij} &\leq 1 \qquad \forall j \in \mathcal{J} \\ \sum_{i=1}^{N_u} \sum_{j=1}^{N_t} x_{ij} &= N_{\min} \triangleq \min\{N_t, N_u L_t\} \\ x_{ij} &\in \{0,1\} \qquad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \end{aligned} \tag{1}$$

where $x_{ij} = 1$ if agent i is assigned to task j and 0 otherwise, and $\mathbf{x}_i \in \{0,1\}^{N_t}$ is a vector whose jth element is x_{ij}. The index sets are defined as $\mathcal{I} \triangleq \{1, \ldots, N_u\}$ and $\mathcal{J} \triangleq \{1, \ldots, N_t\}$. The vector $\mathbf{p}_i \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$ represents an ordered sequence of tasks for agent i; its kth element is $j \in \mathcal{J}$ if agent i conducts j at the kth point along the path, and becomes $\emptyset$ (denoting an empty task) if agent i conducts less than k tasks. The summation term inside the parenthesis represents the local reward for agent i. The *score function* is assumed to satisfy $c_{ij}(\mathbf{x}_i, \mathbf{p}_i) \geq 0$ and can be any nonnegative function of either assignment $\mathbf{x}_i$ or path $\mathbf{p}_i$ (usually not a function of both). In the context of task allocation for autonomous vehicles with mobility, the score function often represents a path-dependent reward, such as the path length, the mission completion time, and the time-discounted value of target.

One special case of interest of the aforementioned formulation is when $L_t = 1$ and $c_{ij}(\mathbf{x}_i, \mathbf{p}_i) \equiv c_{ij}$ without dependency on $\mathbf{x}_i$ and $\mathbf{p}_i$; in this paper, this special case will be called *single-assignment* in contrast to the general *multi-assignment* formulation in (1). The single-assignment problem is important as it can represent a higher level abstraction of a multi-assignment problem with a mathematically simpler form.

This paper will first present an algorithm for the single-assignment case in Section III to provide conceptual insights on the consensus-based auction idea and then extend it to the multi-assignment case in Section IV with a more detailed algorithmic description.

B. Auction Algorithms

One of the key concepts this paper is based on is the auction method for assignment problems. The auction algorithm was first proposed in [26] as a polynomial-time algorithm for the single-assignment problem, and many modifications and extensions have been made to address multi-assignment problems since then. In centralized auction systems [26], the value of a task is given by $c_{ij} = a_{ij} - p_j$, where a_{ij} is the reward of assigning task j to agent i, and p_j is the global price of task j. As the assignment progresses, the value of p_j is continuously updated to reflect the current bid for the task. Auctions are done in rounds and continue until all agents are assigned to the task giving it the maximum value ($\max_j c_{ij}$). Each round selects some agent i that has not been assigned a task and finds out $j^* \triangleq \operatorname{argmax}_j (a_{ij} - p_j)$. If task j^* has already been assigned to another agent, the two agents swap tasks. Once this is done, the price of task j^* is increased such that the value c_{ij^*} is the same as the second highest valued task in agent i's list. Repeating this leads to every agent being assigned to the task giving it the maximum value.

In decentralized methods, the task scores are calculated using $c_{ij} = a_{ij} - p_{ij}$, where p_{ij} is the local price for task j. The bids are generally submitted to an auctioneer [31], [34], [37] to determine the winner based on the highest bids $i^* = \operatorname{argmax}_i c_{ij}$. Other decentralized auction algorithms have been developed that remove the auctioneer in place of different conflict resolution approaches and allow tasks to be bid on asynchronously [50], [51]. The decentralized auction approach developed herein uses a consensus algorithm for conflict resolution without the need for an auctioneer.

C. Consensus Algorithms

For decentralized systems, cooperating agents often require a globally consistent SA [19]. In a dynamic environment with sensor noise and varying network topologies, maintaining consistent SA throughout the fleet can be very difficult. Consensus algorithms are used in these cases to enable the fleet to converge on some specific information set before generating a plan [20]. Examples of typical information sets could be detected target positions, target classifications, and agent positions. These consensus approaches have been shown to guarantee convergence over many different dynamic network topologies [21]–[23].

In this paper, the consensus idea is used to converge on the assignment value rather than the SA. Thus, a *maximum consensus* strategy is implemented such that the current assignment will be overwritten if a higher value is received. By doing this, the network convergence properties found in the consensus algorithm literature can be exploited to converge on the assignment.

III. Consensus-Based Auction Algorithm

The CBAA is a single-assignment strategy that makes use of both auction and consensus. The algorithm consists of iterations between two phases. The first phase of the algorithm is the auction process, while the second is a consensus algorithm that is used to converge on a winning bids list. By iterating between the two, the CBAA can exploit convergence properties of decentralized consensus algorithms as well as the robustness and computational efficiency of the auction algorithms.

Algorithm 1 CBAA Phase 1 for agent i at iteration t

1: **procedure** SELECT TASK($\mathbf{c}_i, \mathbf{x}_i(t-1), \mathbf{y}_i(t-1)$)
2: $\quad \mathbf{x}_i(t) = \mathbf{x}_i(t-1)$
3: $\quad \mathbf{y}_i(t) = \mathbf{y}_i(t-1)$
4: $\quad$ **if** $\sum_j x_{ij}(t) = 0$ **then**
5: $\quad\quad h_{ij} = \mathbb{I}(c_{ij} > y_{ij}(t)),\ \forall j \in \mathcal{J}$
6: $\quad\quad$ **if** $\mathbf{h}_i \neq \mathbf{0}$ **then**
7: $\quad\quad\quad J_i = \operatorname{argmax}_j h_{ij} \cdot c_{ij}$
8: $\quad\quad\quad x_{i,J_i}(t) = 1$
9: $\quad\quad\quad y_{i,J_i}(t) = c_{i,J_i}$
10: $\quad\quad$ **end if**
11: $\quad$ **end if**
12: **end procedure**

A. Phase 1: Auction Process

The first phase of the algorithm is the auction process. Here, each agent places a bid on a task *asynchronously* with the rest of the fleet. Let $c_{ij} \geq 0$ be the bid that agent i places for task j. Two vectors of length N_t that each agent stores and updates throughout the assignment process are also defined. The first vector is $\mathbf{x}_i$, which is agent i's task list, where $x_{ij} = 1$ if agent i has been assigned to task j, and 0 otherwise. The second vector is the winning bids list $\mathbf{y}_i$. This list will be further developed in Section III-B, but it can be assumed for now that y_{ij} is an as up-to-date as possible estimate of the highest bid made for each task thus far. These two vectors are initialized as zero vectors. Using the winning bids list, the list of valid tasks $\mathbf{h}_i$ can be generated using

$$h_{ij} = \mathbb{I}(c_{ij} > y_{ij}) \qquad \forall j \in \mathcal{J} \tag{2}$$

where $\mathbb{I}(\cdot)$ is the indicator function that is unity if the argument is true and zero otherwise.

Algorithm 1 shows the procedure of agent i's phase 1 at iteration t, where one iteration consists of a single run of phase 1 and phase 2. Note that each agent's iteration count can be different, which allows for the possibility that each agent has different iteration periods. An unassigned agent i (equivalently, an agent with $\sum_j x_{ij}(t) = 0$) first computes the valid task list $\mathbf{h}_i$. If there are valid tasks, it then selects a task J_i giving it the maximum score based on the current list of winning bids (line 7 of Algorithm 1), and updates its task $\mathbf{x}_i$ and the winning bids list $\mathbf{y}_i$ accordingly. Also, in the case that the agent has already been assigned a task ($\sum_j x_{ij} \neq 0$), this selection process is skipped, and the agent moves to phase 2.

B. Phase 2: Consensus Process

The second phase of the CBAA is the consensus section of the algorithm. Here, agents make use of a consensus strategy to converge on the list of winning bids and use that list to determine

Algorithm 2 CBAA Phase 2 for agent i at iteration t:

1: SEND $\mathbf{y}_i$ to k with $g_{ik}(\tau)=1$
2: RECEIVE $\mathbf{y}_k$ from k with $g_{ik}(\tau)=1$
3: **procedure** UPDATE TASK($\mathbf{g}_i(\tau)$, $\mathbf{y}_{k\in\{k|g_{ik}(\tau)=1\}}(t)$, J_i)
4: $y_{ij}(t)=\max_k g_{ik}(\tau)\cdot y_{kj}(t),\ \forall j\in\mathcal{J}$
5: $z_{i,J_i}=\text{argmax}_k\, g_{ik}(\tau)\cdot y_{k,J_i}(t)$
6: **if** $z_{i,J_i}\neq i$ **then**
7: $x_{i,J_i}(t)=0$
8: **end if**
9: **end procedure**

the winner. This allows conflict resolution over all tasks while not limiting the network to a specific structure.

Let $\mathbb{G}(\tau)$ be the undirected communication network at time τ with symmetric adjacency matrix $G(\tau)$. The adjacency matrix is defined such that $g_{ik}(\tau)=1$ if a link exists between agents i and k at time τ and 0 otherwise. Agents i and k are said to be *neighbors* if such a link exists. By convention, every node has a self-connected edge; in other words, $g_{ii}(\tau)=1\ \forall i$.

At each iteration of phase 2 of the algorithm, agent i receives the list of winning bids $\mathbf{y}_i$ from each of its neighbors. The procedure of phase 2 is shown in Algorithm 2 when agent i's tth iteration corresponds to τ in real time. The consensus is performed on the winning bids list $\mathbf{y}_i$ based on the winning bids lists received from each neighbor $\mathbf{y}_k$ for all k such that $g_{ik}=1$ in a way that agent i replaces y_{ij} values with the largest value between itself and its neighbors (line 4). Also, an agent loses its assignment if it finds that it is outbid by others for the task it had selected, i.e., $z_{i,J_i}\neq i$ (line 6). Also, this paper assumes that ties occurring in determining J_i in phase 1 or z_{i,J_i} in phase 2 are resolved in a systematic way. For example, a lexicographical tie-breaking heuristic based on the agent and the task identification numbers can be used.

Important properties related to convergence and performance of CBAA will be discussed in Sections V and VI, along with those for a generalized CBAA presented in the following section.

IV. GENERALIZED CBAA: CBBA

As expressed in (1), the scoring function for the multi-assignment problem can depend on the assignment $\mathbf{x}_i$ or the path $\mathbf{p}_i$. To address this dependency, previous combinatorial auction methods [42]–[45] treated each assignment combination (bundle) as a single item for bidding that led to complicated winner selection methods. In this section, CBAA is extended to the multi-assignment problem by presenting the CBBA. In CBBA, each agent has a list of tasks potentially assigned to itself, but the auction process is done at the task level rather than at the bundle level. Similar to CBAA, CBBA consists of iterations between two phases—bundle construction and conflict resolution.

A. Phase 1: Bundle Construction

The first phase of the CBBA algorithm is the bundle-construction process. In contrast to the bundle algorithms in [42]–[45], which enumerate all possible bundles for bidding, in CBBA, each agent creates just a single bundle and updates it as the assignment process progresses. During phase 1 of the algorithm, each agent continuously adds tasks to its bundle until it is incapable of adding any other task. The tasks are added into the bundle in the following way.

Each agent carries two types of lists of tasks: the bundle $\mathbf{b}_i$ and the path $\mathbf{p}_i$. Tasks in the bundle are ordered based on which ones were added first in time, while in the path, they are ordered based on their location in the assignment. Note that the cardinality of $\mathbf{b}_i$ and $\mathbf{p}_i$ cannot be greater than the maximum assignment size L_t. Let $S_i^{\mathbf{p}_i}$ be defined as the total reward value for agent i performing the tasks along the path $\mathbf{p}_i$. In CBBA, if a task j is added to the bundle $\mathbf{b}_i$, it incurs the marginal score improvement of

$$c_{ij}[\mathbf{b}_i]=\begin{cases}0, & \text{if } j\in\mathbf{b}_i\\ \max_{n\leq|\mathbf{p}_i|}S_i^{\mathbf{p}_i\oplus_n\{j\}}-S_i^{\mathbf{p}_i}, & \text{otherwise}\end{cases}\tag{3}$$

where $|\cdot|$ denotes the cardinality of the list, and $\oplus_n$ denotes the operation that inserts the second list right after the nth element of the first list. In the later part of this paper, the notion of $\oplus_{\text{end}}$ will also be used to denote the operation to add the second list at the end of the first one. In other words, the CBBA scoring scheme inserts a new task to the location that incurs the largest score improvement, and this value becomes the marginal score associated with this task given the current path. Thus, if the task is already included in the path, then it provides no additional improvement in score. Also, it is assumed that the addition of any new task provides nontrivial reward, namely, $c_{ij}[\mathbf{b}_i]\geq 0$, and the equality holds only when $j\in\mathbf{b}_i$.

The score function is initialized as $S_i^{\{\emptyset\}}=0$, while the path and bundle are recursively updated as

$$\mathbf{b}_i=\mathbf{b}_i\oplus_{\text{end}}\{J_i\},\qquad \mathbf{p}_i=\mathbf{p}_i\oplus_{n_{i,J_i}}\{J_i\}\tag{4}$$

where $J_i=\text{argmax}_j(c_{ij}[\mathbf{b}_i]\times h_{ij})$, $n_{i,J_i}=\text{argmax}_n S_i^{\mathbf{p}_i\oplus_n\{J_i\}}$, and $h_{ij}=\mathbb{I}(c_{ij}>y_{ij})$. The aforementioned recursion continues until either $|\mathbf{b}_i|=L_t$ or until $\mathbf{h}_i=\mathbf{0}$. Note that with (4), a path is uniquely defined for a given bundle, while multiple bundles might result in the same path.

The first phase of the CBBA is summarized in Algorithm 3. Each agent carries four vectors: a winning bid list $\mathbf{y}_i\in\mathbb{R}_+^{N_t}$, a winning agent list $\mathbf{z}_i\in\mathcal{I}^{N_t}$, a bundle $\mathbf{b}_i\in(\mathcal{J}\cup\{\emptyset\})^{L_t}$, and the corresponding path $\mathbf{p}_i\in(\mathcal{J}\cup\{\emptyset\})^{L_t}$. Note the difference between $\mathbf{x}_i$ used in CBAA and $\mathbf{z}_i$ in CBBA. In CBBA, each agent needs information about not only whether or not it is outbid on the task it selects but who is assigned to each task as well; this enables better assignments based on more sophisticated conflict-resolution rules. These conflict resolution rules are discussed in detail in the following section.

B. Phase 2: Conflict Resolution

In CBAA, agents bid on a single task and release it upon receiving a higher value in the winning bids list. On the contrary, in CBBA, agents add tasks to their bundle based on their currently assigned task set. Suppose that an agent is outbid for

Algorithm 3 CBBA Phase 1 for agent i at iteration t:

procedure BUILD BUNDLE($\mathbf{z}_i(t-1)$, $\mathbf{y}_i(t-1)$, $\mathbf{b}_i(t-1)$)
  $\mathbf{y}_i(t) = \mathbf{y}_i(t-1)$
  $\mathbf{z}_i(t) = \mathbf{z}_i(t-1)$
  $\mathbf{b}_i(t) = \mathbf{b}_i(t-1)$
  $\mathbf{p}_i(t) = \mathbf{p}_i(t-1)$
  while $|\mathbf{b}_i| < L_t$ **do**
    $c_{ij} = \max_{n \leq |\mathbf{p}_i|} S_i^{\mathbf{p}_i \oplus_n \{j\}} - S_i^{\mathbf{p}_i}, \forall j \in \mathcal{J} \setminus \mathbf{b}_i$
    $h_{ij} = \mathbb{I}(c_{ij} > y_{ij}),\ \forall j \in \mathcal{J}$
    $J_i = \operatorname{argmax}_j c_{ij} \cdot h_{ij}$
    $n_{i,J_i} = \operatorname{argmax}_n S_i^{\mathbf{p}_i \oplus_n \{J_i\}}$
    $\mathbf{b}_i = \mathbf{b}_i \oplus_{\text{end}} \{J_i\}$
    $\mathbf{p}_i = \mathbf{p}_i \oplus_{n_{i,J_i}} \{J_i\}$
    $y_{i,J_i}(t) = c_{i,J_i}$
    $z_{i,J_i}(t) = i$
  end while
end procedure

a task and, thus, releases it; then, the marginal score values for the tasks added to the bundle after this task are no longer valid. Therefore, the agent also needs to release all the tasks added after the outbid task. Otherwise, the agent will make further decisions based on wrong score values, which may lead to poor performance.

Releasing the tasks in this manner can, however, cause further complexity in the algorithm. If an agent is able to release tasks without another member selecting it, a simple application of the maximum consensus update on the winning bids list $\mathbf{y}_i$ will no longer converge to the appropriate values since then, the maximum bid observed might no longer be valid. Therefore, the consensus phase of the algorithm needs to be modified in order to ensure that these updates are appropriate.

In the multi-assignment consensus stage, three vectors are communicated for consensus. Two vectors were described in the bundle construction phase: the winning bids list $\mathbf{y}_i \in \mathbb{R}_+^{N_t}$ and the winning agent list $\mathbf{z}_i \in \mathcal{I}^{N_t}$. The third vector $\mathbf{s}_i \in \mathbb{R}^{N_u}$ represents the time stamp of the last information update from each of the other agents. Each time a message is passed, the time vector is populated with

$$s_{ik} = \begin{cases} \tau_r, & \text{if } g_{ik} = 1 \\ \max_{m: g_{im}=1} s_{mk}, & \text{otherwise} \end{cases} \tag{5}$$

where τ_r is the message reception time.

When agent i receives a message from another agent k, $\mathbf{z}_i$ and $\mathbf{s}_i$ are used to determine which agent's information is the most up-to-date for each task. There are three possible actions agent i can take on task j:

1) *update*: $y_{ij} = y_{kj}, z_{ij} = z_{kj}$;
2) *reset*: $y_{ij} = 0, z_{ij} = \emptyset$;
3) *leave*: $y_{ij} = y_{ij}, z_{ij} = z_{ij}$.

Table I outlines the decision rules. The first two columns of the table indicate the agent that each of the sender k and receiver i believes to be the current winner for a given task; the third column indicates the action the receiver should take, where the default action is *leave*.

If a bid is changed by the decision rules in Table I, each agent checks if any of the updated or reset tasks were in their bundle, and if so, these tasks, along with all of the tasks that were added to the bundle after them, are released

$$\begin{aligned} &y_{i,b_{in}} = 0, \qquad z_{i,b_{in}} = \emptyset \qquad \forall n > \bar{n}_i \\ &b_{in} = \emptyset, \qquad n \geq \bar{n}_i \end{aligned} \tag{6}$$

where b_{in} denotes the nth entry of bundle $\mathbf{b}_i$ and $\bar{n}_i = \min\{n : z_{i,b_{in}} \neq i\}$. It should be noted that the wining bid and the winning agent for the tasks added after $b_{i,\bar{n}_i}$ are reset, because removal of $b_{i,\bar{n}_i}$ can change scores for all the ensuing tasks. From here, the algorithm returns to the first phase, and new tasks are added.

Finally, note that CBBA can produce the same solution as CBAA for the problem with $L_t = 1$. The update of the $\mathbf{x}_i$ vector can be equivalently realized by updating $\mathbf{b}_i$, and the conflict-resolution step of the CBAA is equivalent to performing the receiver action rules neglecting $\mathbf{s}_k$ vectors, for only the task that the receiver has selected. Since in CBAA a task is released only when the agent is outbid on that particular task, every incoming $\mathbf{y}_k$ information is valid if it is larger than the local information regardless of the agent belief on when it is sent.

C. Scoring Scheme

1) Diminishing Marginal Gain: One important assumption on the scoring function is that the value of a task does not increase as other elements are added to the set before it. In other words

$$c_{ij}[\mathbf{b}_i] \geq c_{ij}[\mathbf{b}_i \oplus_{\text{end}} \mathbf{b}] \tag{7}$$

for all $\mathbf{b}_i, \mathbf{b}, j$ such that $((\mathbf{b}_i \oplus_{\text{end}} \mathbf{b}) \oplus_{\text{end}} \{j\}) \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$, where $\emptyset$ denotes an empty task. This relation is similar to the notion of submodularity [52] for a set function, except that the bundle is an ordered list rather than an unordered set; this paper will refer to this condition as *diminishing marginal gain* (DMG) and satisfaction of this condition as "being DMG" in the later part. Since the marginal score of task j is defined as (3), the condition (7) can also be expressed in terms of the total score as

$$\begin{aligned} &\max_{n \leq |\mathbf{p}_i|} S_i^{\mathbf{p}_i \oplus_n \{j\}} - S_i^{\mathbf{p}_i} \\ &\quad \geq \max_{n \leq |\mathbf{p}_i|+1} \max_{m \leq |\mathbf{p}_i|} S_i^{(\mathbf{p}_i \oplus_m \{k\}) \oplus_n \{j\}} \\ &\qquad - \max_{m \leq |\mathbf{p}_i|} S_i^{\mathbf{p}_i \oplus_m \{k\}} \end{aligned} \tag{8}$$

for all $\mathbf{p}_i, j, k$ such that $((\mathbf{p}_i \oplus_m \{k\}) \oplus \{j\}) \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$.

It is true that not all of the scoring functions of interest in multi-task allocation satisfy (7). For instance, a scoring scheme with DMG cannot model some synergism by multiple selections. However, in the search and exploration problems for autonomous robots, many reward functions are DMG. For example, in an exploration mission for robotic vehicles, discovery of one feature may provide knowledge about the other targets' locations; thus, the marginal reward of finding other target

TABLE 1
ACTION RULE FOR AGENT i BASED ON COMMUNICATION WITH AGENT k REGARDING TASK j

Agent k (sender) thinks z_{kj} is	**Agent i (receiver) thinks z_{ij} is**	**Receiver's Action (default: leave)**
k	i	if $y_{kj} > y_{ij} \rightarrow$ update
	k	update
	$m \notin \{i, k\}$	if $s_{km} > s_{im}$ or $y_{kj} > y_{ij} \rightarrow$ update
	none	update
i	i	leave
	k	reset
	$m \notin \{i, k\}$	if $s_{km} > s_{im} \rightarrow$ reset
	none	leave
$m \notin \{i, k\}$	i	if $s_{km} > s_{im}$ and $y_{kj} > y_{ij} \rightarrow$ update
	k	if $s_{km} > s_{im} \rightarrow$ update else $\rightarrow$ reset
	m	$s_{km} > s_{im} \rightarrow$ update
	$n \notin \{i, k, m\}$	if $s_{km} > s_{im}$ and $s_{kn} > s_{in} \rightarrow$ update if $s_{km} > s_{im}$ and $y_{kj} > y_{ij} \rightarrow$ update if $s_{kn} > s_{in}$ and $s_{im} > s_{km} \rightarrow$ reset
	none	if $s_{km} > s_{im} \rightarrow$ update
none	i	leave
	k	update
	$m \notin \{i, k\}$	if $s_{km} > s_{im} \rightarrow$ update
	none	leave

decreases. In a time-sensitive target assignment problem, the time-discounted reward for a target decreases as the combat vehicle visits another target first.

In case the scoring scheme is DMG, the following relation is always satisfied:

$$y_{i,b_{in}} \geq y_{i,b_{im}}, \qquad \text{if } n \leq m \tag{9}$$

where b_{ik} is the kth entry of agent i's bundle $\mathbf{b}_i$, because

$$y_{i,b_{in}} = \max_j c_{ij}[\mathbf{b}_i^{1:n-1}] \geq \max_j c_{ij}[\mathbf{b}_i^{1:n-1} \oplus_{\text{end}} \mathbf{b}_i^{n:m-1}] \tag{10}$$

with $\mathbf{b}_i^{k:l} \triangleq \{b_{ik}, \ldots, b_{il}\}$. In other words, the value of y for a task near the start of the bundle is never smaller than that for a task near the end.

2) Time-Discounted Reward: In this paper, the following scoring function representing the time-discounted reward will be considered with specific emphasis [2], [8], [24]:

$$S_i^{\mathbf{p}_i} = \sum \lambda_j^{\tau_i^j(\mathbf{p}_i)} \bar{c}_j \tag{11}$$

where $\lambda_j < 1$ is the discounting factor for task j, $\tau_i^j(\mathbf{p}_i)$ is the estimated time agent i will take to arrive at task location j along the path $\mathbf{p}_i$, and $\bar{c}_j$ is the static score associated with performing task j. The time-discounted reward can model the track scenario in which uncertainty growth with time causes degradation of the expected reward for visiting a certain location or planning of service routes in which satisfaction of client diminishes with time. Since the triangular inequality holds for the actual distance between task locations

$$\tau_i^j(\mathbf{p}_i \oplus_n \{k\}) \geq \tau_i^j(\mathbf{p}_i) \qquad \forall n \quad \forall k. \tag{12}$$

In other words, if an agent moves along a longer path, then it arrives at each of the task locations at later time than if it moves along a shorter path, thus resulting in further discounted score value. Thus, for all nonnegative constants $\bar{c}_j$'s, $S_i^{\mathbf{p}_i}$ in (11) is DMG.

V. CONVERGENCE

This section analyzes the convergence properties of CBBA, where convergence means producing an assignment in finite time with all of the constraints in (1) being satisfied.

A. Sequential Greedy Algorithm

This section starts by presenting a centralized algorithm that will be shown to give the same solution as CBBA gives, in Section V-B. Consider the *sequential greedy algorithm* (SGA) in Algorithm 4 that sequentially finds a sequence of agent–task pairs that render the largest score values given prior selections. This algorithm is a centralized procedure in the sense that a single central agent can access every agent's scoring scheme; every agent's scoring scheme is assumed to be DMG. Note that if $\eta_i < L_t$, the score update in lines 16 and 17 of Algorithm 4

Algorithm 4 Sequential greedy algorithm

1: $\mathcal{I}_1 = \mathcal{I},\ \mathcal{J}_1 = \mathcal{J}$
2: $\eta_i = 0,\ \forall i \in \mathcal{I}$
3: $c_{ij}^{(1)} = c_{ij}[\{\emptyset\}],\ \forall (i,j) \in \mathcal{I} \times \mathcal{J}$
4: **for** $n = 1$ to $N_{\min}$ **do**
5: $\quad (i_n^\star, j_n^\star) = \text{argmax}_{(i,j)\in\mathcal{I}\times\mathcal{J}}\, c_{ij}^{(n)}$
6: $\quad \eta_{i_n^\star} = \eta_{i_n^\star} + 1$
7: $\quad \mathcal{J}_{n+1} = \mathcal{J}_n \setminus \{j_n^\star\}$
8: $\quad \mathbf{b}_{i_n^\star}^{(n)} = \mathbf{b}_{i_n^\star}^{(n-1)} \oplus_{\text{end}} \{j_n^\star\}$
9: $\quad \mathbf{b}_i^{(n)} = \mathbf{b}_i^{(n-1)},\ \forall i \neq i_n^\star$
10: $\quad$ **if** $\eta_{i_n^\star} = L_t$ **then**
11: $\quad\quad \mathcal{I}_{n+1} = \mathcal{I}_n \setminus \{i_n^\star\}$
12: $\quad\quad c_{i_n^\star, j}^{(n+1)} = 0, \forall j \in \mathcal{J}$
13: $\quad$ **else**
14: $\quad\quad \mathcal{I}_{n+1} = \mathcal{I}_n$
15: $\quad$ **end if**
16: $\quad c_{i,j_n^\star}^{(n+1)} = 0,\ \forall i \in \mathcal{I}_{n+1}$
17: $\quad c_{ij}^{(n+1)} = c_{ij}[\mathbf{b}_i^{(n)}],\ \forall (i,j) \in \mathcal{I}_{n+1} \times \mathcal{J}_{n+1}$
18: **end for**

results in

$$c_{ij}^{(n+1)} = \begin{cases} c_{ij}^{(n)}, & \text{if } i \neq i_n^\star,\ j \neq j_n^\star \\ 0, & \text{if } j = j_n^\star \\ \alpha_{ij}^{(n)} c_{ij}^{(n)}, & \text{if } i = i_n^\star,\ j \neq j_n^\star \end{cases} \tag{13}$$

with some $\alpha_{ij}^{(n)} \in [0,1]$, because $\mathbf{b}_i^{(n)}$ remains the same for $i \neq i_n^\star$, and the marginal gains that $i_n^\star$ can achieve diminish as one task is added in its bundle. In case $\eta_i = L_t$ for some agent i, all of the agent's scores for the next selection step become zero (line 12). Thus, for $\eta_i \leq L_t$, the score $c_{ij}^{(n)}$ is *monotonically decreasing* with respect to n, namely

$$c_{ij}^{(n)} \geq c_{ij}^{(m)}, \qquad \text{if } n \leq m. \tag{14}$$

Also, by definition of DMG

$$c_{ij}^{(n)} = c_{ij}[\mathbf{b}_i^{(n-1)}] \geq c_{ij}[\mathbf{b}_i^{(n-1)} \oplus_{\text{end}} \mathbf{b}] \quad \forall \mathbf{b} \tag{15}$$

which means that $c_{ij}^{(n)}$ is the largest score agent i can obtain for task j given prior selection of $\mathbf{b}_i^{(n-1)}$. Since the selected pair at the nth step, $(i_n^\star, j_n^\star)$ in line 5, gives the largest score given selections up to the $(n-1)$th step, the following is satisfied:

$$c_{i_n^\star, j_n^\star}^{(n)} \geq c_{ij}^{(n)} \qquad \forall (i,j) \in \mathcal{I} \times \mathcal{J}. \tag{16}$$

Therefore, note that from (14) and (16)

$$c_{i_n^\star, j_n^\star}^{(n)} \geq c_{i_m^\star, j_m^\star}^{(n)} \geq c_{i_m^\star, j_m^\star}^{(m)} \geq c_{ij}^{(m)} \qquad \forall (i,j) \in \mathcal{I} \times \mathcal{J} \quad \text{if } n \leq m. \tag{17}$$

Namely, the best score at the nth step is greater than, or equal to, any score value showing up in the later steps. It is also noted that the recursion in (13) leads to

$$c_{ij}^{(n+1)} = 0 \qquad \forall (i,j) \notin \mathcal{I}_{n+1} \times \mathcal{J}_{n+1} \tag{18}$$

because the marginal score of a task that is already in a bundle is zero.

B. Static Network

The communication network of a fleet of unmanned vehicles can be modeled as an undirected graph with every edge length being unity. Suppose that this communication network is static and connected; then, there exists a (undirected) shortest path length $d_{ik} < \infty$ for every pair of agents i and k. The network diameter D is defined as the longest of all shortest path lengths

$$D \triangleq \max_{(i,k)\in\mathcal{I}^2} d_{ik}. \tag{19}$$

If the conflict resolution is assumed to be *synchronized*, i.e., every agent's second phase in the tth iteration takes place simultaneously, then the actual time τ can be equivalently represented by the iteration count t. In this case, the convergence time $T_C \in \mathbb{Z}_+$ can be defined as the smallest iteration number at which a feasible assignment is found that will not change afterwards

$$T_C \triangleq \min t \in \mathcal{T} \tag{20}$$

where the set $\mathcal{T}$ is defined as

$$\mathcal{T} = \Bigg\{ t \in \mathbb{Z}_+ \Big| \forall s \geq t : x_{ij}(s) = x_{ij}(t), \sum_{i=1}^{N_u} x_{ij}(s) = 1 \sum_{j=1}^{N_t} x_{ij}(s) \leq L_t, \sum_{j=1}^{N_t}\sum_{i=1}^{N_u} x_{ij}(s) = N_{\min} \Bigg\} \tag{21}$$

with x_{ij} being the same binary variable defined in (1).

Lemma 1: Consider the CBBA process with synchronous conflict resolution over a static network with diameter D for the case that every agent's scoring scheme is DMG. Suppose that after completing phase 2 of some iteration t

$$z_{i,j_k^\star}(t) = i_k^\star, y_{i,j_k^\star}(t) = c_{i_k^\star, j_k^\star}^{(k)} \qquad \forall i \in \mathcal{I} \qquad \forall k \leq n \tag{22}$$

where $(i_k^\star, j_k^\star)$'s are assignment pairs from the SGA procedure and $c_{i_k^\star, j_k^\star}^{(k)}$'s are the corresponding score values. Then, the following holds.

1) The first $L_i^{(n)} \triangleq |\mathbf{b}_i^{(n)}|$ entries of agent i's current bundle coincide with those of the bundle at the nth SGA step $\mathbf{b}_i^{(n)}$

$$\mathbf{b}_i^{1:L_i^{(n)}} = \mathbf{b}_i^{(n)}. \tag{23}$$

2) The bid that agent $i_{n+1}^\star$ places on task $j_{n+1}^\star$ is

$$y_{i_{n+1}^\star, j_{n+1}^\star}(t) = c_{i_{n+1}^\star, j_{n+1}^\star}^{(n+1)} \tag{24}$$

and this value satisfies

$$y_{i_{n+1}^\star, j_{n+1}^\star}(t) \geq y_{ij}(t) \qquad \forall (i,j) \in \mathcal{I}_{n+1} \times \mathcal{J}_{n+1}. \tag{25}$$

3) Entries in (22) do not change over time, or

$$z_{i,j_k^\star}(s) = z_{i,j_k^\star}(t), \qquad y_{i,j_k^\star}(s) = y_{i,j_k^\star}(t) \tag{26}$$

for all $s \geq t$ and for all $k \leq n$.

4) The value of the bid that agent $i_{n+1}^\star$ places on task $j_{n+1}^\star$ will remain the same throughout the later iterations, and

no agents will bid higher than this value on task $j^\star_{n+1}$ in the later iterations:

$$y_{i^\star_{n+1},j^\star_{n+1}}(s) = y_{i^\star_{n+1},j^\star_{n+1}}(t) \geq y_{i,j^\star_{n+1}}(s) \qquad (27)$$

for all $s \geq t$ and for all $i \in \mathcal{I}$.

5) After D iterations, every agent will have agreed on the assignment $(i^\star_{n+1}, j^\star_{n+1})$; in other words

$$y_{i,j^\star_{n+1}}(t+D) = y_{i^\star_{n+1},j^\star_{n+1}}(t) \quad z_{i,j^\star_{n+1}}(t+D) = i^\star_{n+1} \qquad (28)$$

for all $i \in \mathcal{I}$.

Proof: See Appendix A. ∎

Lemma 2: Consider a CBBA process with synchronized conflict resolution over a static network of diameter D, where every agent's scoring scheme is DMG. Then, every agent agrees on the first n SGA assignments by iteration nD. In other words,

$$z_{i,j^\star_k}(nD) = i^\star_k \qquad \forall i \in \mathcal{I} \qquad \forall k \leq n \qquad (29)$$

$$y_{i,j^\star_k}(nD) = c^{(k)}_{i^\star_k,j^\star_k} \qquad \forall i \in \mathcal{I} \qquad \forall k \leq n. \qquad (30)$$

Proof: The proof is by induction. Since $\operatorname{argmax}_{j\in\mathcal{J}} c_{i^\star_1,j}[\{\emptyset\}] = j^\star_1$, agent $i^\star_1$ places task $j^\star_1$ in the first position of its bundle in phase 1 of iteration 1. Because $c_{i^\star_1,j^\star_1}[\{\emptyset\}] \geq c_{ij}[\mathbf{b}]$ for all $(i,j) \in \mathcal{I}\times\mathcal{J}$ for any $\mathbf{b}$, no agent can place a higher bid in later iterations. Thus, k iterations of CBBA conflict resolution procedures lead k-hop neighbors of agent $i^\star_1$ to agree that $i^\star_1$ is the winning agent for task $j^\star_1$. Thus, after D iterations of phase 2, every agent will have agreed on the assignment $(i^\star_1, j^\star_1)$. Due to statements 3) and 5) in Lemma 1, if $z_{i,j^\star_k}(mD) = i^\star_k$ and $y_{i,j^\star_k}(mD) = c^{(k)}_{i^\star_k,j^\star_k}$ for all $k \leq m$, then $z_{i,j^\star_k}(mD+D) = i^\star_k$ and $y_{i,j^\star_k}(mD+D) = c^{(k)}_{i^\star_k,j^\star_k}$ for all $k \leq m+1$. Thus, together with $(i^\star_1, j^\star_1)$ being agreed to at D, after nD iterations, every agent will have agreed on the assignments $(i^\star_k, j^\star_k)$ for all $k \leq n$. ∎

Theorem 1 (Convergence of CBBA): Provided that the scoring function is DMG, the CBBA process with a synchronized conflict resolution phase over a static communication network with diameter D satisfies the following.

1) CBBA produces the same solution as SGA with the corresponding winning bid values and winning agent information being shared across the fleet, i.e.,

$$z_{i,j^\star_k} = i^\star_k \qquad \forall k \leq N_{\min} \quad \forall i \in \mathcal{I}$$

$$y_{i,j^\star_k} = c^{(k)}_{i^\star_k,j^\star_k} \qquad \forall k \leq N_{\min} \quad \forall i \in \mathcal{I}. \qquad (31)$$

2) The convergence time T_C is bounded above by $N_{\min}D$.

Proof: Combining the statement 5) in Lemma 1 and Lemma 2, after phase 2 of iteration nD, the first n SGA assignments are agreed over the fleet for any n. This must be true in case $n = N_{\min}$, which is the number of assignments needed for convergence. In addition, from statement 3), these assignments will not change in the later iterations. Thus, by iteration $N_{\min}D$, the CBBA process converges, and the converged solutions are equivalent to the SGA solution. ∎

Note that in many cases, CBBA converges much earlier than $N_{\min}D$ iterations, because the maximum distance from some $i^\star_k$ to another agent is likely to be less than D, and multiple SGA assignment sequences can be fixed simultaneously. Quick convergence of CBBA will be empirically verified in Section VII-A.

Lemma 3: Suppose that the score values of the agents satisfy

$$c_{ij}(t) \geq c_{ij}(s) \qquad \forall (i,j) \in \mathcal{I}\times\mathcal{J} \qquad \forall t \leq s \qquad (32)$$

$$c_{ij}(t) \geq c_{kj}(t) \Leftrightarrow c_{ij}(s) \geq c_{kj}(s) \qquad \forall i,k \in \mathcal{I} \qquad \forall j \in \mathcal{J} \qquad \forall t,s \in \mathbb{N} \qquad (33)$$

in the process of CBBA, where $c_{ij}(t)$ is agent i's score for task j at iteration t. Then, CBBA converges to a conflict-free assignment in $N_{\min}D$ iterations in a static network with diameter D.

Proof: The key idea of the proof is to consider a sequential procedure that is similar to SGA but replaces line 5 in Algorithm 4 by

$$(i^\dagger_n, j^\dagger_n) = \underset{(i,j)\in\mathcal{I}\times\mathcal{J}}{\operatorname{argmax}} c_{ij}(t^\dagger_n)$$

where $t^\dagger_n \triangleq (n-1)D+1$. Then, to show that $y_{i^\dagger_n,j^\dagger_n}$ is not outbid in the later iterations completes the proof of this lemma. See Appendix B for details. ∎

The conditions in Lemma 3 can be restrictive. For example, the time-discounted reward in (11), which is DMG, does not satisfy them in general. However, Lemma 3 facilitates modification of CBBA to render a conflict-free assignment, even when the scoring schemes are not DMG.

Lemma 4: Consider a CBBA process with synchronized conflict resolution over a static network of diameter D, where agents' scoring schemes are *not necessarily* DMG. Let $c^o_{ij}(t)$ be the score of task j for agent i computed by this underlying scoring scheme. Then, CBBA converges to a conflict-free assignment within $N_{\min}D$ by utilizing the following modified score instead of $c^o_{ij}(t)$:

$$c_{ij}(t) = \min\left\{c^o_{ij}(t), c_{ij}(t-1)\right\}. \qquad (34)$$

Proof: The proof is straightforward, since (34) ensures (32) and (33). ∎

To summarize, in a static network with diameter D, CBBA (with a slight modification) creates a conflict-free assignment within $N_{\min}D$ iterations independent of scoring schemes. Moreover, if the score function is DMG, it generates the identical solution to SGA.

C. Dynamic Network and Asynchronous Conflict Resolution

For dynamic networks in which $G(\tau)$ varies with time, convergence of CBBA with a synchronous conflict resolution phase can still be guaranteed if there exists some value $\rho < \infty$ such that

$$\mathbb{W}(\tau(t)) = \mathbb{G}(\tau(t)) \cup \mathbb{G}(\tau(t+1)) \cup \cdots \cup \mathbb{G}(\tau(t+\rho-1))$$

is fully connected $\forall t$ [53], where $\tau(t)$ denotes the actual time at which every agent's tth CBBA iteration takes place. In this case, the convergence time will then be upper bounded by $\rho N_{\min}$, since any information about conflicts is transmitted within ρ.

Asynchronous conflict resolution can be modeled as a dynamic network with synchronized conflict resolution, as the

situation where an agent is waiting for neighbors' information can be treated as the network being disconnected for that period. Thus, if it is ensured that an agent eventually communicates with its neighbor, then the CBBA process converges in finite time, even in the case when asynchronous conflict resolution is allowed.

D. Inconsistent Information

It is typical that each agent's scoring scheme is based on its own understanding of the environment (which is commonly known as the SA). For instance, the time-discounted reward in (11) depends on the target and agent locations; therefore, with different estimates of either, the resulting scores used by the agents in CBBA may differ. Since these scores may also differ from the (typically not knowable) actual scores, the CBBA solution based on inconsistent information over fleet can degrade the performance of the decision-making process.

However, this inconsistency in SA does not affect the convergence of CBBA to a feasible assignment, because whatever knowledge each agent scoring scheme is based on, the only needed information for resolving conflicts among agents are the winning bid list, winning agent list, and the time stamp. If these three pieces of information are communicated error-free, the conflict resolution process of CBBA is insensitive to the details of each agent's scoring scheme. Thus, CBBA does not require any level of agreement on SA for convergence, although inconsistent information might still cause actual performance degradation. This is a distinguishing feature of CBBA compared with previous decentralized algorithms such as implicit coordination [8], [14] and the ETSP ASSIGMT algorithm [50], [51], in which each agent must have the same information to guarantee convergence.

VI. MINIMUM PERFORMANCE GUARANTEE

This section shows that the CBBA and CBAA solutions guarantee some performance level without considering the actual scoring scheme. First, define the following quantities:

1) $SOPT$: the optimal objective value of the single-assignment problem for a given nonnegative scoring scheme;
2) $CBAA$: the objective value provided by CBAA for the single-assignment problem for a given nonnegative scoring scheme;
3) $MOPT$: the optimal objective value of the multi-assignment problem for a given nonnegative DMG scoring scheme;
4) $CBBA$: the objective value provided by CBBA for the multi-assignment problem for a given nonnegative DMG scoring scheme.

The worst-case performance analysis addresses the relationship between $MOPT$ and $CBBA$ (or between $SOPT$ and $CBAA$). This section starts with the single-assignment case.

Lemma 5 (CBAA performance bound): Assuming the agents have accurate knowledge of the SA, CBBA guarantees 50% optimality. In other words

$$SOPT \leq 2CBAA. \tag{35}$$

Proof: Since the CBBA solution provides the same performance as SGA, it is sufficient to prove that the SGA solution guarantees 50% optimality. First, for notational convenience, reorder the agent and target indices so that

$$i_k^\star = k, \qquad j_k^\star = k \qquad \forall k \leq N_{\min}. \tag{36}$$

In other words, for now, refer to agent $i_k^\star$ as agent k and task $j_k^\star$ as task k, while other indices are adjusted accordingly to avoid overlap. Then, from the property in (17), for SGA

$$c_{ii} \geq c_{jj}, \qquad \text{if } i < j \tag{37}$$

and the objective value of CBAA solution (or equivalently SGA solution) becomes

$$CBAA = \sum_{i=1}^{N_{\min}} c_{ii}. \tag{38}$$

Because each agent selects its task in a greedy way given the selections of its precedents, the following inequalities hold for the greedy solution:

$$\begin{aligned} c_{ii} &\geq c_{ij} \qquad \forall i \qquad \forall j > i \\ c_{ii} &\geq c_{ji} \qquad \forall i \qquad \forall j > i. \end{aligned} \tag{39}$$

Consider the case the greedy selection is the farthest from the optimal solution; in other words, consider of the case where variations of assignment could cause the largest improvement in the objective value while still satisfying the conditions in (39). Also, since each agent cannot take multiple tasks, a change in the assignment should be based on swapping of the tasks (or possibly cyclic exchange of tasks). Consider a task swapping between two agents i and $j > i$; then, the overall score becomes $c_{ij} + c_{ji}$, while it was originally $c_{ii} + c_{jj}$. Since (39) holds, the new overall score $c_{ij} + c_{ji}$ is upper bounded by

$$c_{ij} + c_{ji} \leq c_{ii} + c_{ii} = 2c_{ii} \tag{40}$$

where the upper bound is attained if

$$c_{ij} = c_{ji} = c_{ii}. \tag{41}$$

Thus, if (41) holds, agents i and j can increase their overall score the most by swapping their tasks. Now, supposing that the condition similar to (41) holds for all pairs of agents

$$\begin{aligned} c_{ij} &= c_{ii} \qquad \forall i \qquad \forall j > i \\ c_{ji} &= c_{ii} \qquad \forall i \qquad \forall j > i \end{aligned} \tag{42}$$

then an appropriate sequence of task swapping processes will lead to the largest possible improvement of the overall score among the fleet.

One way to achieve the greatest performance enhancement is to use the following policy:

$$J_i^* = \begin{cases} N_{\min} - i + 1, & \text{if } i \in \{1, \ldots, N_{\min}\} \\ \emptyset, & \text{otherwise} \end{cases} \tag{43}$$

where J_i^* is the new task assigned to agent i, in which agent $i \in \{1, \ldots, N_{\min}\}$ swaps its task with agent $N_{\min} - i + 1$. In this way, the first $\lceil N_{\min}/2 \rceil$ agents (who were assigned tasks by CBAA) are assigned tasks that provide the same scores as the CBAA solution, while the next $\lfloor N_{\min}/2 \rfloor$ agents (who were assigned tasks by CBAA) gain as much as possible score improvement. Since the policy in (43) ensures that one agent is assigned at most one task, it creates a conflict-free assignment; moreover, as the overall score is improved as much as it can be, the resulting solution is the optimal solution. Hence, the optimal objective value $SOPT$ should satisfy

$$\begin{aligned} SOPT &= \sum_{i=1}^{\lceil N_{\min}/2 \rceil} c_{ii} + \sum_{i=\lceil N_{\min}/2 \rceil + 1}^{N_{\min}} c_{(N_{\min}-i+1),(N_{\min}-i+1)} \\ &= 2 \times \sum_{i=1}^{\lfloor N_{\min}/2 \rfloor} c_{ii} + \sum_{i=\lfloor N_{\min}/2 \rfloor + 1}^{\lceil N_{\min}/2 \rceil} c_{ii} \leq 2 \times \sum_{i=1}^{N_{\min}} c_{ii} \\ &= 2CBAA. \end{aligned}$$

Thus, 50% optimality is guaranteed for the CBAA. ∎

Based on the aforementioned proof for the CBAA solution for single-assignment problems, the worst-case performance bound for the CBBA solution for multi-assignment problems can also be derived.

Theorem 2 (CBBA performance bound): Assuming the agents have accurate knowledge of the SA, CBBA guarantees 50% optimality for the multi-assignment problem with DMG scoring schemes

$$MOPT \leq 2\, CBBA. \tag{44}$$

Proof: The key idea of the proof is that a multi-assignment problem can be treated as a single-assignment problem with additional combinatorial number of virtual agents. See Appendix C for the detailed proof. ∎

Note that in many cases, CBBA creates a numerical solution providing much greater than 50% optimality. This observation is consistent with the very good average performance of CBBA that is analytically demonstrated for several special cases in [1].

VII. Numerical Results

A. Convergence and Performance With Inconsistent Information

As discussed in Section V-D, the presented CBBA method guarantees convergence of the algorithm to a conflict-free assignment, regardless of inconsistency in SA. Monte-Carlo simulations are performed to verify this robustness property. The agents and tasks are randomly placed on a $W \times W$ 2-D space ($W = 2$ km). The time-discounted reward in (11) is used to define the scoring function. $\bar{c}_j \equiv 1$ and $\lambda = 0.95$ s^{-1} are used, and every agent moves at a speed of 40 m/s.

The source of inconsistent information considered is discrepancy in the understanding of task locations, while it is assumed that each agent knows its own position correctly. Agents estimate the coordinates of task locations subject to additive Gaussian noises (with sensing noise standard deviation from $0.01W$ to $0.2W$) and compute the score values based on these estimates. As a metric of level of inconsistency across the fleet, the following average SA error is calculated:

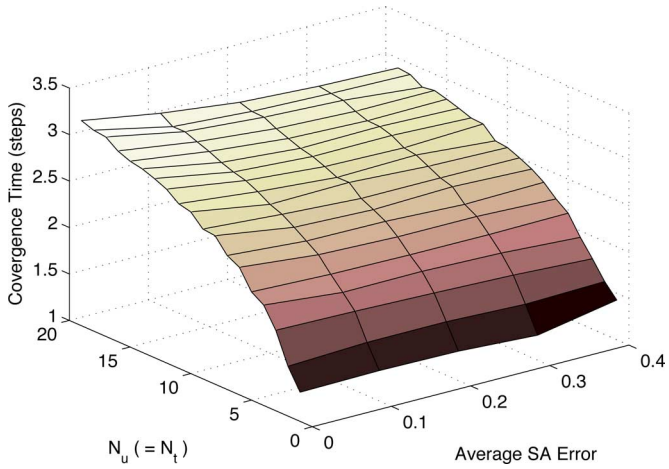


Fig. 1. Convergence of CBBA in the presence of inconsistency in SA.

$$\bar{E}_{\mathrm{SA}} = \frac{1}{\sqrt{2} W N_t} \sqrt{\sum_{i,k:i\neq k}^{N_u} \sum_{j=1}^{N_t} \|\mathbf{l}_{ij} - \mathbf{l}_{kj}\|^2} \tag{45}$$

where $\mathbf{l}_{ij}$ and $\mathbf{l}_{kj}$ are the estimated position vectors of task j by agent i and k, respectively, and $\|\cdot\|$ denotes the Euclidian norm. Each agent computes the scores for the tasks based on its own estimate of target positions. Communication networks are created by generating a random spanning tree [54] and then adding varying amounts of random links to the network. Also, the optimal solution with perfect information is obtained by the implicit coordination algorithm [8] for comparison.

Figs. 1 and 2 show the average convergence time and the optimality gap as a function of N_t, which is set to be same as N_u in this simulation, and the SA error. Note in Fig. 1 that SA error does not affect the convergence time of CBBA, as the algorithm converges within a few time steps for all cases. The optimality gap in Fig. 2 demonstrates that (a) with perfect information, the optimality gap is very small (less than 3%), (b) performance of CBBA degrades as SA error grows, and (c) but, even with a large amount of SA error, the CBBA solution exhibits reasonable good average performance (optimality gap being less than 30%). To summarize, the results verify that CBBA produces a reasonably good suboptimal solution, even with a significant amount of discrepancy in the SA.

B. Comparison With Prim Allocation for Multi-Assignment

For further validation of the convergence and performance aspects of CBBA, this paper compares CBBA with an existing centralized sequential auction algorithm, i.e., Prim Allocation (PA) [33]. The PA algorithm is a well-known auction algorithm for multi-assignment and has a similar insertion heuristic to the score definition in (3), thus providing a good baseline for comparison.

In the PA algorithm, each agent creates a minimum spanning tree (MST) with the tasks as nodes and the edges indicating the

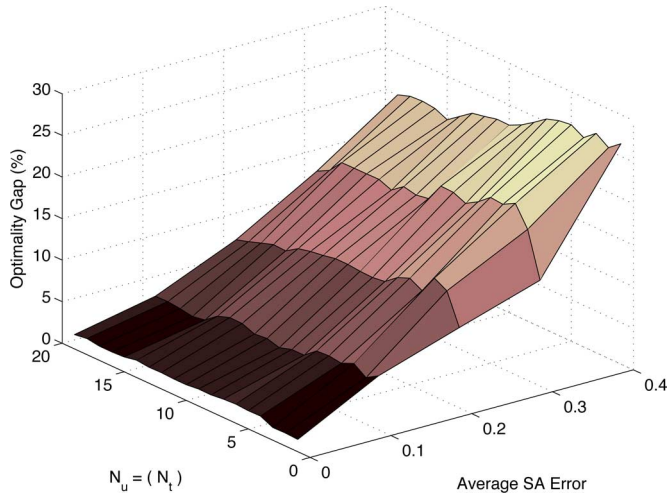


Fig. 2. Optimality gap of CBBA in the presence of inconsistency in SA.

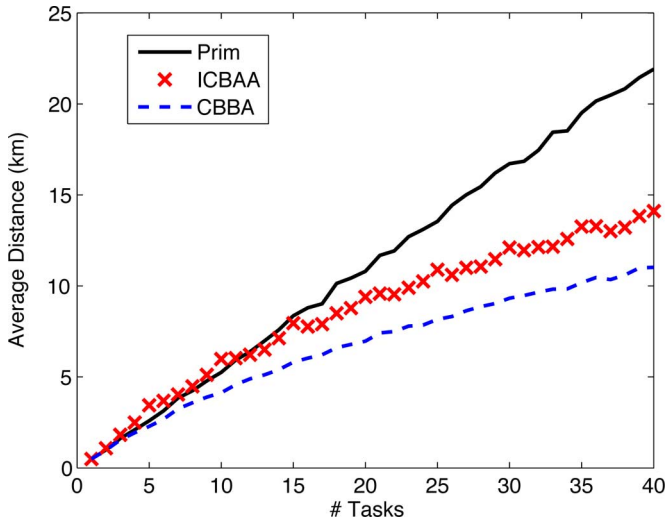


Fig. 3. Total distance traved to accomplish assignment ($N_u = 5$).

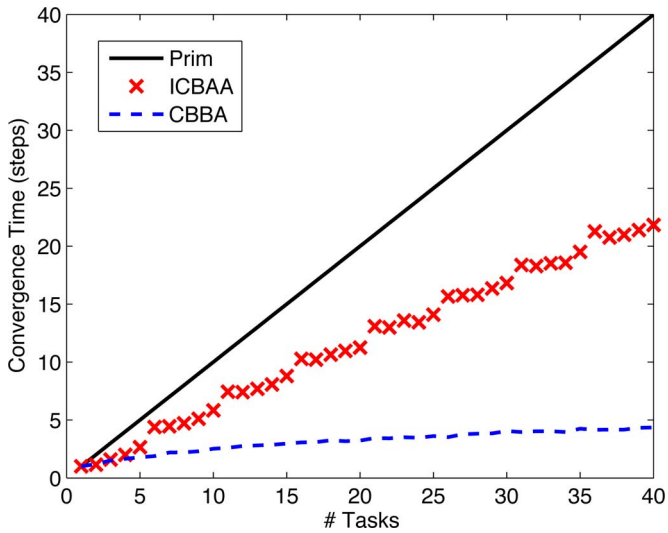


Fig. 4. Convergence time steps ($N_u = 5$).

task ordering. Each agent bids on the task that is closest to any of the nodes in the assignment, and the winner inserts it into that location in the tree. This process continues until all of the tasks have been assigned. Tasks are then ordered through the tree by performing a depth-first search (DFS) [55]. The algorithm is designed to minimize the total distance traveled by the fleet to accomplish the tasks; however, other heuristics have been developed in [56] and [57] that can be used as well.

The iterative CBAA (ICBAA) that sequentially runs CBAA single-assignment routine until all the tasks are assigned is also considered for comparison. A key difference between ICBAA and CBBA solutions is that the same number of tasks are assigned per agent in ICBAA, while CBBA allows an agent to take up to L_t tasks.

The total distances traveled by the vehicles for the PA algorithm and CBBA are compared with each other. CBBA tries to maximize the time-discounted reward in (11) instead of minimizing the total distance traveled. This is a good example of how the use of generic scores when developing algorithms can provide more flexibility in terms of objective functions. Fig. 3 compares the performance of the three algorithms for different N_t values, where $N_u = 5$ and $L_t = N_t$ are used. It can be seen that CBBA provides a solution with the smallest total distance traveled, although it does not explicitly minimize it. This is because the CBBA algorithm is able to outbid earlier allocated tasks in the conflict resolution stage to provide better assignments. For the PA, once a task has a winner, it is locked into that assignment. The convergence times for the three algorithms are compared in Fig. 4; to account for centralized aspect of the PA algorithm, a fully connected network (i.e., $D = 1$) is assumed for ICBAA and CBBA. Since the PA algorithm assigns each task one at a time, the convergence time steps for PA is same as the number of tasks in a fully connected network. It can be found that ICBAA converges within about a half of the number of tasks, and CBBA converges within much smaller steps. Since ICBAA consists of $\lceil N_t/N_u \rceil$ individual CBAA routines, the total convergence time step is approximately $\lceil N_t/N_u \rceil$ multiplied by convergence time of a single CBAA. A single CBAA can take up to $\min\{N_u, N_t\}D$ iterations by Theorem 1, but usually takes much shorter because multiple conflicts can be resolved per time step. As a result, ICBAA converges faster than PA by the factor of the number of conflicts resolved in parallel in CBAA. For CBBA, multiple tasks are assigned to an agent, and multiple conflicts are resolved in parallel, because each agent carries a bundle of tasks that can be very long if L_t is large. These two capabilities of CBBA enable further acceleration.

VIII. Conclusion

This paper presented two decentralized task-allocation algorithms addressing single- and multi-assignment problems, respectively, that are shown to produce conflict-free solutions independent of inconsistencies in SA. These algorithms feature a task-selection process based on auctioning with greedy heuristics and a conflict-resolution protocol based on consensus on the winning bid values over the team. It was also shown that the

solutions for the proposed algorithms guarantee 50% of optimality under the assumptions of consistent SA and DMG from tasks. Numerical experiments validated good performance and quick convergence of the proposed methods compared with an existing sequential auction algorithm.

APPENDIX

A. Proof of Lemma 1

Statement 1): The proof is by induction. Suppose that $\mathbf{b}_i^{(n)} = \{j^\star_{k_1}, \ldots, j^\star_{k_{L_i^{(n)}}}\}$ with some $k_1 < \cdots < k_{L_i^{(n)}}$, where $L_i^{(n)} \triangleq |\mathbf{b}_i^{(n)}|$. Then, the first entry $j^\star_{k_1}$ is determined from

$$c_{i,j^\star_{k_1}}^{(k_1)} = c_{i,j^\star_{k_1}}[\{\emptyset\}] = \max_{j \in \mathcal{J}_{k_1}} c_{i,j}[\{\emptyset\}] \tag{46}$$

where $\{\emptyset\}$ denotes the null bundle, and $\mathcal{J}_{k_1}$ is the reduced task set defined from the recursion in line 7 in Algorithm 4 with $n+1 = k_1$ (or by line 1 for $k_1 = 1$), because no task has been selected in advance of $j^\star_{k_1}$. On the other hand, the phase 1 of the CBBA process for agent i finds the first entry in the bundle by

$$J_i = \operatorname*{argmax}_{j \in \mathcal{J}} c_{i,j}[\{\emptyset\}] \times \mathbb{I}\left(c_{ij}[\{\emptyset\}] > y_{ij}\right) \tag{47}$$

where $\mathbb{I}(\cdot)$ is the indicator function that is unity if the argument is satisfied and zero otherwise. Note that $c_{ij}[\{\emptyset\}] \leq y_{ij}$ for $j \notin \mathcal{J}_{k_1}$, because all these j's are assigned to the other agents. Thus, the maximization in (47) is equivalent to the maximization in (46) that searches over a more restricted set $\mathcal{J}_{k_1}$. Hence, J_i in (47) equals to $j^\star_{k_1}$; the corresponding score values are identical: $c_{i,J_i} = c_{i,j^\star_{k_1}}^{(k_1)}$. Thus, the first entry of the CBBA bundle is $j^\star_{k_1}$, which is also the first entry of the SGA bundle.

Now suppose that the SGA bundle and the CBBA bundle coincide up to the lth entry. Then, the $(l+1)$th entry of the SGA bundle $j^\star_{k_{l+1}}$ is determined from

$$c_{i,j^\star_{k_{l+1}}}^{(k_{l+1})} = c_{i,j^\star_{k_{l+1}}}[\mathbf{b}_i^{(1:l)}] = \max_{j \in \mathcal{J}_{k_{l+1}}} c_{i,j}[\mathbf{b}_i^{(1:l)}] \tag{48}$$

where $\mathbf{b}_i^{(1:l)}$ represents the list of the first l entries of agent i's SGA bundle. Consider a task $j \notin \mathcal{J}_{k_{l+1}}$; then, either of the following is the case: $j \in \mathbf{b}_i^{(1:l)}$ or $j \in (\mathcal{J} \setminus \mathcal{J}_{k_{l+1}}) \setminus \mathbf{b}_i^{(1:l)}$. If $j \in \mathbf{b}_i^{(1:l)}$, then $c_{ij}[\mathbf{b}_i^{(1:l)}] = 0$ from (3). Otherwise, $c_{ij}[\mathbf{b}_i^{(1:l)}] \leq y_{ij}$, because task j must then be in another agent's bundle. Thus, the following holds:

$$c_{i,j}[\mathbf{b}_i^{(1:l)}] \times \mathbb{I}\left(c_{ij}[\mathbf{b}_i^{(1:l)}] > y_{ij}\right) = 0, \qquad j \notin \mathcal{J}_{k_{l+1}} \tag{49}$$

either because the first term is zero (for $j \in \mathbf{b}_i^{(1:l)}$) or the second term is zero (for $j \in (\mathcal{J} \setminus \mathcal{J}_{k_{l+1}}) \setminus \mathbf{b}_i^{(1:l)}$). On the other hand, the corresponding entry of the CBBA bundle of agent i is determined from

$$\max_{j \in \mathcal{J}} c_{i,j}[\mathbf{b}_i^{(1:l)}] \times \mathbb{I}\left(c_{ij}[\mathbf{b}_i^{(1:l)}] > y_{ij}\right). \tag{50}$$

Using the result in (49)

$$\begin{aligned} &\max_{j \in \mathcal{J}} c_{i,j}[\mathbf{b}_i^{(1:l)}] \times \mathbb{I}\left(c_{ij}[\mathbf{b}_i^{(1:l)}] > y_{ij}\right) \\ &= \max_{j \in \mathcal{J}_{k_{l+1}}} c_{i,j}[\mathbf{b}_i^{(1:l)}] \times \mathbb{I}\left(c_{ij}[\mathbf{b}_i^{(1:l)}] > y_{ij}\right) \\ &= \max_{j \in \mathcal{J}_{k_{l+1}}} c_{i,j}[\mathbf{b}_i^{(1:l)}] \end{aligned} \tag{51}$$

since every score value is nonnegative. Note that (51) is identical to the maximization in (48). Thus, if the first l entries of the SGA and the CBBA bundles coincide, the $(l+1)$th entries also coincide because they are computed from two equivalent procedures. Together with the coincidence of the first entry, this completes the proof showing that $\mathbf{b}_i^{1:L_i^{(n)}} = \mathbf{b}_i^{(n)}$.

Statement 2): The proof is in two parts. First, it is proved that at iteration t, agent $i^\star_{n+1}$ places a bid of $c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}}$ on task $j^\star_{n+1}$, where $(i^\star_{n+1}, j^\star_{n+1})$ is the $(n+1)$th SGA agent–task pair. Second, it is proved that this bid is greater than any bid for agent $i \in \mathcal{I}_{n+1}$ on task $j \in \mathcal{J}_{n+1}$.

In the $(n+1)$th step of SGA procedure, agent $i^\star_{n+1}$ determines the corresponding task from the following maximization:

$$\max_{j \in \mathcal{J}_{n+1}} c^{(n+1)}_{i^\star_{n+1},j}. \tag{52}$$

Since statement 1) in this lemma holds for agent $i^\star_{n+1}$, the $(L^{(n)}_{i^\star_{n+1}} + 1)$th entry of its CBBA bundle is selected from

$$\max_{j \in \mathcal{J}} c_{i^\star_{n+1},j}[\mathbf{b}^{(n)}_{i^\star_{n+1}}] \times \mathbb{I}\left(c_{i^\star_{n+1},j}[\mathbf{b}^{(n)}_{i^\star_{n+1}}] > y_{i^\star_{n+1},j}\right). \tag{53}$$

Note that for $j \notin \mathcal{J}_{n+1}$, either of the following holds: 1) $c_{i^\star_{n+1},j}[\mathbf{b}^{(n)}_{i^\star_{n+1}}] = 0$ for $j \in \mathbf{b}^{(n)}_{i^\star_{n+1}}$, or 2) $c_{i^\star_{n+1},j}[\mathbf{b}^{(n)}_{i^\star_{n+1}}] \leq y_{i^\star_{n+1},j}$ for $j \notin \mathbf{b}^{(n)}_{i^\star_{n+1}}$. Thus, the maximization in (53) is equivalent to the maximization in (52) because $c^{(n+1)}_{i^\star_{n+1},j} = c_{i^\star_{n+1},j}[\mathbf{b}^{(n)}_{i^\star_{n+1}}]$. Hence, $i^\star_{n+1}$ places a bid of $c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}}$ on task $j^\star_{n+1}$ and locates it at the $(L_i^{(n)} + 1)$th position of its CBBA bundle; also, the corresponding entry of its winning bid list is set as

$$y_{i^\star_{n+1},j^\star_{n+1}}(t) = c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}}. \tag{54}$$

The second part is to prove that

$$c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}} \geq y_{ij}(t) \qquad \forall (i,j) \in \mathcal{I}_{n+1} \times \mathcal{J}_{n+1}. \tag{55}$$

Consider the maximization to determine the $(n+1)$th SGA selection; then, the following relation holds:

$$c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}} \triangleq \max_{(i,j) \in \mathcal{I} \times \mathcal{J}} c^{(n+1)}_{ij} = \max_{(i,j) \in \mathcal{I}_{n+1} \times \mathcal{J}_{n+1}} c_{ij}[\mathbf{b}^{(n)}_i] \tag{56}$$

since $c^{(n+1)}_{ij} = 0$ for $(i,j) \notin \mathcal{I}_{n+1} \times \mathcal{J}_{n+1}$ by (18). Because statement 1) specifically holds for $i \in \mathcal{I}_{n+1}$

$$c_{ij}[\mathbf{b}_i^{1:L_i^{(n)}}(t)] = c_{ij}[\mathbf{b}_i^{(n)}]$$

where $\mathbf{b}_i(t)$ is agent i's CBBA bundle at iteration t. Note that for $(i,j) \in \mathcal{I}_{n+1} \times \mathcal{J}_{n+1}$, the winning bid value $y_{ij}(t)$

$$y_{ij}(t) = c_{k,j}[\mathbf{b}_k^{1:L_k^{(n)}}(t) \oplus_{\text{end}} \mathbf{b}] \tag{57}$$

with some $k \in \mathcal{I}_{n+1}$ and $\mathbf{b}$ such that $\mathbf{b}_k^{1:L_k^{(n)}}(t) \oplus_{\text{end}} \mathbf{b} \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$. Since the scoring scheme is assumed to be DMG, this leads to

$$y_{ij}(t) \leq c_{k,j}[\mathbf{b}_k^{1:L_k^{(n)}}] = c_{k,j}[\mathbf{b}_k^{(n)}] \tag{58}$$

for the same k as in (57). From (56) and (58), it follows that

$$c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}} \geq y_{ij}(t) \qquad \forall (i,j) \in \mathcal{I}_{n+1} \times \mathcal{J}_{n+1}.$$

Statement 3): The proof is by induction. First, for the score value of the first SGA assignment determined by

$$c^{(1)}_{i^\star_1,j^\star_1} = c_{i^\star_1,j^\star_1}[\{\emptyset\}] = \max_{(i,j)\in\mathcal{I}\times\mathcal{J}} c_{ij}[\{\emptyset\}]$$

the following holds:

$$c^{(1)}_{i^\star_1,j^\star_1} = c_{i^\star_1,j^\star_1}[\{\emptyset\}] \geq c_{i,j^\star_1}[\mathbf{b}] \quad \forall i \in \mathcal{I} \quad \forall \mathbf{b} \in (\mathcal{J} \cup \{\emptyset\})^{L_t}. \tag{59}$$

Note that

$$\begin{aligned} &\forall y_{ij}(r), \qquad (i,j) \in \mathcal{I} \times \mathcal{J}, \qquad r \geq 1 \\ &\exists k \in \mathcal{I}, \qquad \mathbf{b} \in (\mathcal{J} \cup \{\emptyset\})^{L_t}, \quad \text{such that } y_{ij}(r) = c_{kj}[\mathbf{b}]. \end{aligned} \tag{60}$$

Specifically, at $r = t$ for any i and for $j = j^\star_1$, $k = i^\star_1$ and $\mathbf{b} = \{\emptyset\}$ satisfy (60)

$$y_{i,j^\star_1}(t) = c_{i^\star_1,j^\star_1}[\{\emptyset\}]$$

where $y_{i,j^\star_1}(t)$ can be changed in the later iteration only when some agent places a bid larger than it, but (59) prevents the occurrence of such situations. Therefore

$$y_{i,j^\star_1}(s) = y_{i,j^\star_1}(t) = c^{(1)}_{i^\star_1,j^\star_1} \qquad \forall i \in \mathcal{I} \qquad \forall s \geq t \tag{61}$$

which also means $z_{i,j^\star_1}(s) = i^\star_1 \; \forall i \in \mathcal{I} \; \forall s \geq t$.

Now, suppose that $y_{i,j^\star_k}(s) = y_{i,j^\star_k}(t) = c^{(k)}_{i^\star_k,j^\star_k} \; \forall s \geq t$ for all $k \leq m$ for some $m < n$. Then, since statement 1) with n being replaced by m holds at iteration s, $\mathbf{b}_i^{1:|\mathbf{b}_i^{(m)}|}(s) = \mathbf{b}_i^{(m)}$. Consider the $(|\mathbf{b}^{(m)}_{i^\star_{m+1}}| + 1)$th entry of agent $i^\star_{m+1}$'s CBBA bundle. From statement 2) in this lemma, the entry is $\{j^\star_{m+1}\}$, and the corresponding bid is $c^{(m+1)}_{i^\star_{m+1},j^\star_{m+1}}$, which satisfies

$$c^{(m+1)}_{i^\star_{m+1},j^\star_{m+1}} \geq c_{ij}[\mathbf{b}_i^{1:|\mathbf{b}_i^{(m)}|} \oplus_{\text{end}} \mathbf{b}] \tag{62}$$

for all $(i,j) \in \mathcal{I} \times \mathcal{J}$ for any $\mathbf{b} \in (\mathcal{J} \cup \{\emptyset\})^{L_t - |\mathbf{b}_i^{(m)}|}$.

Since it is assumed in (22) that $y_{i,j^\star_{m+1}}(t) = c^{(m+1)}_{i^\star_{m+1},j^\star_{m+1}}$, the following identity holds:

$$y_{i^\star_{m+1},j^\star_{m+1}}(s) = c^{(m+1)}_{i^\star_{m+1},j^\star_{m+1}} = y_{i,j^\star_{m+1}}(t) \qquad \forall i. \tag{63}$$

The CBBA conflict resolution does not replace a winning bid unless a higher bid shows up; for $y_{i,j^\star_{m+1}}(t)$, no agent can place a higher bid between iteration t and s because of (62). Thus, if $y_{i,j^\star_k}(s) = y_{i,j^\star_k}(t) = c^{(k)}_{i^\star_k,j^\star_k} \; \forall s \geq t$ for all $k \leq m$ for some $m < n$, then $y_{i,j^\star_{m+1}}(s) = y_{i,j^\star_{m+1}}(t) = c^{(m+1)}_{i^\star_{m+1},j^\star_{m+1}}$. Together with (61), this completes the proof using induction.

Statement 4): From statement 3) in this lemma, at any iteration $s \geq t$, (22) is satisfied. From statement 2) in this lemma, this means $y_{i^\star_{n+1},j^\star_{n+1}}(s) = c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}} \geq y_{ij}(s) \; \forall (i,j) \in \mathcal{I}_{n+1} \times \mathcal{J}_{n+1}$ for any $s \geq t$. Moreover, with statement 3) being satisfied, agent $i^\star_{n+1}$ will not change its bid on $j^\star_{n+1}$ after iteration t; thus, $y_{i^\star_{n+1},j^\star_{n+1}}(s) = y_{i^\star_{n+1},j^\star_{n+1}}(t)$.

Statement 5): Because $c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}}$ is the highest bid on task $j^\star_{n+1}$ for all $s \geq t$, the conflict resolution phase of CBBA leads to $i \neq i^\star_{n+1}$ updating $y_{i,j^\star_{n+1}}$ with $c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}}$. Since the agents in agent $i^\star_{n+1}$'s k-hop neighbors perform this update in k iterations from t, and the farthest agent from $i^\star_{n+1}$ is at most D hops apart, every agent will have agreed on the winning bid on task $j^\star_{n+1}$ by iteration $t + D$.

B. Proof of Lemma 3

Consider a procedure that samples the score values at every $t^\dagger_n$th iteration of the CBBA process, where $t^\dagger_n \triangleq (n-1)D + 1, n \in \mathbb{N}$, and solves the following maximization:

$$(i^\dagger_n, j^\dagger_n) = \operatorname*{argmax}_{(i,j)\in\mathcal{I}\times\mathcal{J}} c_{ij}(t^\dagger_n). \tag{64}$$

Then, the following holds:

$$y_{i^\dagger_n,j^\dagger_n}(t^\dagger_n) \geq y_{i,j^\dagger_n}(t) \qquad \forall i \in \mathcal{I} \qquad \forall t \geq t^\dagger_n \tag{65}$$

which is proved as follows. By definition, $c_{i^\dagger_n,j^\dagger_n}(t^\dagger_n) \geq c_{i,j^\dagger_n}(t^\dagger_n) \; \forall i \in \mathcal{I}^\dagger_n$; the condition (33) ensures that $c_{i^\dagger_n,j^\dagger_n}(t) \geq c_{i,j^\dagger_n}(t) \; \forall t \in \mathbb{N}$. Since at every iteration agent $i^\dagger_n$ achieves the largest score on task $j^\dagger_n$, its respective bid is the largest at every iteration: $y_{i^\dagger_n,j^\dagger_n}(t) \geq y_{i,j^\dagger_n}(t) \; \forall i \in \mathcal{I} \; \forall t \in \mathbb{N}$; this specifically means that (65) is satisfied.

Since no agent can bid higher than $y_{i^\dagger_n,j^\dagger_n}(t^\dagger_n)$ on task $j^\ddagger_n$ at any iteration later than $t^\dagger_n$, this winning bid information is propagated through the entire network within D iterations: $y_{i,j^\dagger_n}(t^\dagger_{n+1}) = y_{i^\dagger_n,j^\dagger_n}(t^\dagger_n)$. Therefore, it is straightforward to show that at $N_{\min}D$

$$\begin{aligned} &y_{i,j^\dagger_n}(N_{\min}D) = y_{i^\dagger_n,j^\dagger_n}(t^\dagger_n) \qquad \text{and} \\ &z_{i,j^\dagger_n}(N_{\min}D) = i^\dagger_n \; \forall i \in \mathcal{I}. \end{aligned}$$

C. Proof of Theorem 2

The multi-assignment problem can be treated as a single assignment with additional combinatorial number of agents. Let agent $i^{\mathbf{b}}, \mathbf{b} \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$ be a *virtual* agent that can be

assigned at most one task and whose score is defined in such a way that

$$c_{i^{\mathbf{b}},j} = c_{ij}[\mathbf{b}].$$

Then, there will be a total of $N_u^M \triangleq N_u \sum_{n=1}^{N_t} N_t!/n!$ agents (because the bundle is an ordered list not unordered set), each of which can only select up to one task. Consider a single-assignment problem for these artificial agents, and call it *expanded* single-assignment.

Since a task already in a bundle incurs zero reward and the scoring schemes are assumed to be DMG, the scores for the expanded single assignment should satisfy

$$\begin{aligned} &c_{i^{\mathbf{b}},j} = 0, \qquad \text{if } j \in \mathbf{b} \\ &c_{i^{\mathbf{b}_1},j} \geq c_{i^{\mathbf{b}_1 \oplus_{\text{end}} \mathbf{b}_2},j} \qquad \forall j \in \mathcal{J} \qquad \forall \mathbf{b}_1, \mathbf{b}_2. \end{aligned} \tag{66}$$

Similar to (35) in Lemma 5, the agent and task indices can be reordered such that

$$i_k^{\star\,\mathbf{b}_{i_k^\star}^{(k-1)}} = k, \qquad j_k^\star = k \qquad \forall k \leq N_{\min} \triangleq \min\{N_u L_t, N_t\}. \tag{67}$$

For these reordered virtual agents and tasks, the objective value for the CBBA solution becomes

$$CBBA = \sum_{i=1}^{N_{\min}} c_{ii} \tag{68}$$

with the following conditions being satisfied:

$$\begin{aligned} &c_{ii} \geq c_{kk}, \qquad \text{for } k > i \\ &c_{ij} \leq c_{ii}, \qquad c_{ji} \leq c_{ii}, \qquad \text{for } j > i. \end{aligned} \tag{69}$$

Now, consider a task swapping procedure for optimality. Similar to Lemma 5, the high-ranked agent tries to choose a task with smallest loss, while the low-ranked agent tries to pick a task with the highest gain. However, for this expanded single-assignment case, the task swapping process is more restricted than the case in Lemma 5, because agent $i^{\{\emptyset\}}$ and agent $i^{\mathbf{b}}$ (both in terms of indices before reordering) cannot independently select their tasks. For instance, supposing that agent $i^{\{\emptyset\}}$, who has been assigned task j, picks another task j', then agent $i^{\{j\}}$ must release its assignment. Thus, the reselection process is not simply based on pairwise (or cyclic) task swappings. However, it should be noted that the optimal solution obtained by considering all these restrictions is bounded above by the *unconstrained swapping* solution that allows inadmissible task swapping as if the expanded single-assignment is identical to a single-assignment problem.

There is still another restriction in performing this unconstrained task swapping: It should be ensured that $c_{i_k^\star,\mathbf{b}',j_k^\star} = 0$ for $\mathbf{b}' = \mathbf{b}_{i_k^\star}^{(k-1)} \oplus_{\text{end}} \mathbf{b}$ with $\mathbf{b} \neq \{\emptyset\}$, while the swapping policy in (43) leads to $c_{i_k^\star,\mathbf{b}',j_k^\star} = c_{i_k^\star,j_k^\star}^{(k)}$. However, note that the maximum achievable score increases if this restriction is relaxed, and a swapping policy similar to (43) renders the maximum achievable score for this relaxation. Thus, $MOPT$ is upper bounded by the score generated by the policy in (43) applied to the expanded single assignment

$$\begin{aligned} MOPT &\leq \sum_{i=1}^{\lceil N_{\min}/2 \rceil} c_{ii} + \sum_{i=\lceil N_{\min}/2 \rceil + 1}^{N_{\min}} c_{(N_{\min}-i+1),(N_{\min}-i+1)} \\ &= 2 \times \sum_{i=1}^{\lfloor N_{\min}/2 \rfloor} c_{ii} + \sum_{i=\lfloor N_{\min}/2 \rfloor + 1}^{\lceil N_{\min}/2 \rceil} c_{ii} \\ &\leq 2 \times \sum_{i=1}^{N_{\min}} c_{ii} = 2\, CBBA. \end{aligned} \tag{70}$$

Therefore, CBBA guarantees 50% optimality.

ACKNOWLEDGMENT

The authors thank Dr. M. Alighanbari for his invaluable contribution in the development of the precursors to the CBAA.

REFERENCES

[1] L. Brunet, H.-L. Choi, and J. P. How, "Consensus-based auction approaches for decentralized task assignment," presented at the AIAA Guid., Navigat., Control Conf., Honolulu, HI, 2008, vol. AIAA-2008-6839.

[2] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperative UAVs," in *Cooperative Control: Models, Applications, and Algorithms*, S. Butenko, R. Murphey, and P. M. Pardalos, Eds. Boston, MA: Kluwer, 2003, pp. 23–41.

[3] C. Schumacher, P. Chandler, and S. Rasmussen, "Task allocation for wide area search munitions," in *Proc. Amer. Control Conf.*, 2002, pp. 1917–1922.

[4] C. Cassandras and W. Li, "A receding horizon approach for solving some cooperative control problems," in *Proc. IEEE Conf. Decis. Control*, 2002, pp. 3760–3765.

[5] Y. Jin, A. Minai, and M. Polycarpou, "Cooperative real-time search and task allocation in UAV teams," in *Proc. IEEE Conf. Decis. Control*, 2003, pp. 7–12.

[6] L. Xu and U. Ozguner, "Battle management for unmanned aerial vehicles," in *Proc. IEEE Conf. Decis. Control*, 2003, pp. 3585–3590.

[7] D. Turra, L. Pollini, and M. Innocenti, "Fast unmanned vehicles task allocation with moving targets," in *Proc. IEEE Conf. Decis. Control*, Dec. 2004, pp. 4280–4285.

[8] M. Alighanbari, "Task assignment algorithms for teams of UAVs in dynamic environments," Master's thesis, Mass. Inst. Technol., Cambridge, MA, 2004.

[9] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative-timing missions," *J. Guid., Control, Dyn.*, vol. 28, no. 1, pp. 150–161, 2005.

[10] D. Castanon and C. Wu, "Distributed algorithms for dynamic reassignment," in *Proc. IEEE Conf. Decis. Control*, 2003, pp. 13–18.

[11] J. Curtis and R. Murphey, "Simultaneous area search and task assignment for a team of cooperative agents," presented at the AIAA Guid., Navigat., Control Conf. Exhib., Austin, TX, 2003.

[12] T. Shima, S. J. Rasmussen, and P. Chandler, "UAV team decision and control using efficient collaborative estimation," in *Proc. Amer. Control Conf.*, 2005, pp. 4107–4112.

[13] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, Sep. 2004.

[14] W. Ren, R. Beard, and D. Kingston, "Multi-agent Kalman consensus with relative uncertainty," in *Proc. Amer. Control Conf.*, 2005, pp. 1865–1870.

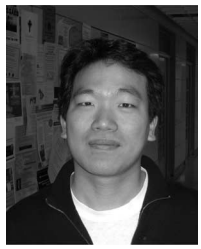
[15] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[16] M. Alighanbari and J. P. How, "An unbiased Kalman consensus algorithm," in *Proc. Amer. Control Conf.*, 2006, pp. 3519–3524.

[17] C. C. Moallemi and B. V. Roy, "Consensus propagation," *IEEE Trans. Inf. Theory*, vol. 52, no. 11, pp. 4753–4766, Nov. 2006.

[18] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," in *Proc. 45th IEEE Conf. Decis. Control*, 2006, pp. 3387–3392.

[19] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle control," *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 71–82, Apr. 2007.
[20] R. Beard and V. Stepanyan, "Synchronization of information in distributed multiple vehicle coordinated control," in *Proc. IEEE Conf. Decis. Control*, 2003, pp. 2029–2034.
[21] Y. Hatano and M. Mesbahi, "Agreement over random networks," in *Proc. 43rd IEEE Conf. Decis. Control*, 2004, pp. 2010–2015.
[22] C. W. Wu, "Synchronization and convergence of linear dynamics in random directed networks," *IEEE Trans. Autom. Control*, vol. 51, no. 7, pp. 1207–1210, Jul. 2006.
[23] A. Tahbaz-Salehi and A. Jadbabaie, "On consensus over random networks," in *Proc. 44th Annu. Allerton Conf.*, 2006, pp. 1315–1321.
[24] M. Alighanbari and J. P. How, "Decentralized task assignment for unmanned aerial vehicles," in *Proc. 44th IEEE Conf. Decis. Control, Eur. Control Conf.*, 2005, pp. 5668–5673.
[25] D. Dionne and C. A. Rabbath, "Multi-UAV decentralized task allocation with intermittent communications: The DTC algorithm," in *Proc. Amer. Control Conf.*, 2007, pp. 5406–5411.
[26] D. P. Bertsekas, "The auction algorithm for assignment and other network flow problems," Mass. Inst. Technol., Cambridge, MA, Tech. Rep., 1989.
[27] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. IEEE*, vol. 94, no. 7, pp. 1257–1270, Jul. 2006.
[28] B. Gerkey and M. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 758–768, Oct. 2002.
[29] D. P. Bertsekas, "Auction algorithms" in *Encyclopedia of Optimization*. Norwell, MA: Kluwer, 2001.
[30] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, 2004.
[31] A. M. Kwasnica, J. O. Ledyard, D. Porter, and C. DeMartini, "A new and improved design for multiobject iterative auctions," *Manage. Sci.*, vol. 51, no. 3, pp. 419–434, 2005.
[32] P. Milgrom, "Putting auction theory to work: The simultaneous ascending auction," *J. Political Econ.*, vol. 108, no. 2, pp. 245–272, 2000.
[33] M. G. Lagoudakis, M. Berhaultt, S. Koenigt, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multirobot task allocation," in *Proc. IEEE/RSI Int. Conf. Intell. Robots Syst.*, 2004, pp. 698–705.
[34] S. Sariel and T. Balch, "Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments," in *Proc. AAAI Workshop: Integrating Planning Into Scheduling*, 2005, pp. 27–33.
[35] A. Ahmed, A. Patel, T. Brown, M. Ham, M. Jang, and G. Agha, "Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism," in *Proc. Int. Conf. Integr. Knowl. Intensive Multi-Agent Syst.*, 2005, pp. 311–317.
[36] M. L. Atkinson, "Results analysis of using free market auctions to distribute control of UAVs," presented at the AIAA 3rd "Unmanned Unlimited" Tech. Conf., Workshop Exhib., Chicago, IL, 2004.
[37] T. Lemaire, R. Alami, and S. Lacroix, "A distributed task allocation scheme in multi-UAV context," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 3622–3627.
[38] W. Walsh and M. Wellman, "A market protocol for decentralized task allocation," in *Proc. Int. Conf. Multi Agent Syst.*, 1998, pp. 325–332.
[39] M. Hoeing, P. Dasgupta, P. Petrov, and S. O'Hara, "Auction-based multirobot task allocation in COMSTAR," in *Proc. 6th Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2007, pp. 1–8.
[40] P. B. Sujit and R. Beard, "Distributed sequential auctions for multiple UAV task allocation," in *Proc. Amer. Control Conf.*, 2007, pp. 3955–3960.
[41] X. Zheng, S. Koenig, and C. Tovey, "Improving sequential single-item auctions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 2238–2244.
[42] D. C. Parkes and L. H. Ungar, "Iterative combinatorial auctions: Theory and practice," in *Proc. 17th Nat. Conf. Artif. Intell.*, 2000, pp. 74–81.
[43] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2003, pp. 1957–1962.
[44] A. Andersson, M. Tenhunen, and F. Ygge, "Integer programming for combinatorial auction winner determination," presented at the Fourth Int. Conf. MultiAgent Syst., Boston, MA, 2000.
[45] S. de Vries and R. Vohra, "Combinatorial auctions: A survey," *INFORMS J. Comput.*, vol. 15, no. 3, pp. 284–309, 2003.
[46] M. H. Rothkopf, A. Pekec, and R. M. Harstad, "Computationally manageable combinatorial auctions," Rutgers Univ., New Brunswick, NJ, Tech. Rep., 1998.
[47] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," *Artif. Intell.*, vol. 135, no. 1/2, pp. 1–54, 2002.
[48] M. Nandy and A. Mahanti, "An improved search technique for optimal winner determination in combinatorial auctions," in *Proc. 37th Hawaii Int. Conf. Syst. Sci.*, 2004, p. 10.
[49] M. Mito and S. Fujita, "On heuristics for solving winner determination problem in combinatorial auctions," *J. Heuristics*, vol. 10, no. 5, pp. 507–523, 2004.
[50] S. L. Smith and F. Bullo, "Target assignment for robotic networks: Asymptotic performance under limited communication," in *Proc. Amer. Control Conf.*, 2007, pp. 1155–1160.
[51] S. L. Smith and F. Bullo, "Monotonic target assignment for robotic networks," *IEEE Trans. Autom. Control*, vol. 54, no. 10, 2009.
[52] S. Fujishige, "Submodular functions optimization," in *Annals of Discrete Mathematics*. Amsterdam, The Netherlands: Elsevier, 1991.
[53] V. D. Blondel, J. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. 44th IEEE Conf. Decis. Control, Eur. Control Conf.*, 2005, pp. 2996–3000.
[54] D. B. Wilson, "Generating random spanning trees more quickly than the cover time," in *Proc. 28th Annu. ACM Symp. Theory Comput.*, 1996, pp. 296–303.
[55] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, T. M. Press, Ed., Cambridge, MA: MIT Press, 1990.
[56] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain, "Auction-based multirobot routing," in *Proc. Robot.: Sci. Syst.*, 2005, pp. 343–350.
[57] C. Tovey, M. G. Lagoudakis, S. Jain, and S. Koenig, "The generation of bidding rules for auction-based robot coordination," in *Proc. 3rd Int. Multi-Robot Syst. Workshop*, 2005, pp. 3–14.



Han-Lim Choi (S'07–M'08) received the B.S. and M.S. degrees in aerospace engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2000 and 2002, respectively, and the Ph.D. degree in aeronautics and astronautics from Massachusetts Institute of Technology (MIT), Cambridge, in 2008.

He is currently a Postdoctoral Associate with the Department of Aeronautics and Astronautics, MIT. His current research interests include estimation and control for sensor networks and decision making for multiagent systems.

Dr. Choi is a member of the American Institute of Aeronautics and Astronautics.

Luc Brunet received the B.Eng. degree (with high distinction) in aerospace engineering from Carleton University, Ottawa, ON, Canada, in 2006 and the M.Sc. degree in aeronautics and astronautics from Massachusetts Institute of Technology, Cambridge, in 2008.

He is currently a Collaborative Robotics Specialist at Frontline Robotics, Ottawa.

Jonathan P. How (S'90–M'92–SM'05) received the B.A.Sc. degree from the University of Toronto, Toronto, ON, Canada, in 1987 and the S.M. and Ph.D. degrees in aeronautics and astronautics from Massachusetts Institute of Technology (MIT), Cambridge, in 1990 and 1993, respectively.

He joined MIT in 2000, where he was a Postdoctoral Associate for two years for the Middeck Active Control Experiment (MACE) and is currently a Professor with the Department of Aeronautics and Astronautics. He was an Assistant Professor with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA. His current research interests include robust coordination and control of autonomous vehicles in dynamic uncertain environments.

Prof. How was the recipient of the 2002 Institute of Navigation Burka Award. He is an Associate Fellow of the American Institute of Aeronautics and Astronautics.