AIAA Guidance, Navigation and Control Conference and Exhibit
18 - 21 August 2008, Honolulu, Hawaii

AIAA 2008-6839

Consensus-Based Auction Approaches for Decentralized Task Assignment

Luc Brunet*, Han-Lim Choi†, and Jonathan P. How‡
Massachusetts Institute of Technology, Cambridge, MA 02139

This paper addresses task assignment in the coordination of a fleet of unmanned vehicles by presenting two decentralized algorithms: consensus-based auction algorithm (CBAA) and its generalization to the multi-assignment problem, 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 by achieving agreement on the winning bid values. The conflict resolution process of CBBA is further enhanced to address the dependency of the score value on the previously selected tasks in the multi-assignment setting. This work shows that the proposed algorithms, under reasonable assumptions on the scoring scheme and network connectivity, guarantee convergence to a conflict-free assignment. Also, the converged solutions are shown to guarantee 50% optimality in the worst-case and to exhibit provably good performance on average. Moreover, the proposed algorithms produce a feasible assignment even in the presence of inconsistency in situational awareness across the fleet, and even when the score functions varies with time in some standard manner. Numerical experiments verify quick convergence and good performance of the presented methods for both static and dynamic assignment problems.

I. Introduction

Cooperation amongst a fleet of robotic agents is necessary in order to improve the overall performance of most missions. Many different methods exist that enable a group of such agents to allocate tasks amongst themselves from a known list. Centralized planners[1–7] 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, making the robots smaller and cheaper to build. On the other hand, agents must consistently communicate with a fixed location, reducing the possible mission ranges that the fleet can handle, as well as creating a single point of failure in the mission.

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[8–11]. 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[12]. 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[13–18] to converge on a consistent SA before performing the assignment[19]. These consensus algorithms can guarantee convergence of the SA over many different dynamic network topologies[20–22], 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[23]. This can cause severe latency in low

*S.M. Candidate, Dept. Aeronautics and Astronautics at the time of submission; currently with Frontline Robotics, Ottawa, Ontario, Canada, lbrunet@mit.edu

†PhD Candidate, Dept. of Aeronautics and Astronautics, hanlimc@mit.edu, Student Member AIAA, (Corresponding Author).

‡Professor, Dept. of Aeronautics and Astronautics, jhow@mit.edu, Associate Fellow AIAA.

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: the robust decetralized task asslignment (RDTA) algorithm[23] enhanced robustness to inconsistent SA by allowing agents to communicate plans as well as SA, while the decentralized task consensus (DTC) algorithm[24] restricted communication occurrence only to the cases there is mismatch between plans based on the local knowledge and on the estimated global knowledge[24]. 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[25–28] are another method for task assignment that have been shown to efficiently produce near-optimal solutions in terms of communication and computation[29]. 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 act as the auctioneer to receive and evaluate each bid in the fleet[30–32]. Once all of the bids have been collected, a winner is selected based on a pre-defined scoring metric. In other formulations, the central system is removed and one of the bidders acts as the auctioneer[33–37]. 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 solutions 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. Similar to the RDTA algorithm, 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[38,39].

In summary, algorithms that use consensus before planning tend to exhibit more robustness to temporal variation of network topologies, while traditional auction approaches tend to be computationally efficient and robust to inconsistencies in the SA. This paper presents algorithms that aim to combine both approaches in order to take advantage of properties from both assignment strategies. The paper will first present the Consensus-Based Auction Algorithm (CBAA), which is a single-assignment greedy strategy that makes use of a consensus algorithm for conflict resolution. The CBAA will then be extended to the more general Consensus-Based Bundle Algorithm (CBBA) for multiple assignment.

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[33,39,40]. Bundle approaches[41–44] 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 of enumerating all possible bundle combinations, and winner determination has been shown to be **NP**-complete[45], requiring the use of specialized winner determination algorithms[46–48] for computational tractability.

CBBA, however, builds a single bundle and bids on the included tasks based on the improvement they provide to the bundle. Computational effort is reduced by considering only a single bundle while convergence times are improved over sequential auctions since multiple tasks can be assigned in parallel. The algorithm will be shown to guarantee convergence to a conflict-free solution under some mild assumptions on the scoring schemes and the network connectivity. Under the same assumptions, this work will also show that the converged solution of CBBA represents provably good performance. Numerical simulations with UAV-target assignment problems with time-discounted rewards will validate the convergence and performance properties of the proposed methods. The capability of handling time-varying aspects of the score functions will also be empirically verified, in particular, compared with an existing dynamic target assignment algorithm[49,50].

II. Background

II.A. Task Assignment Problems

The goal of task assignment problem 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 above 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:

$$\begin{aligned} \max \quad & \sum_{i=1}^{N_u} \left(\sum_{j=1}^{N_t} c_{ij}(\mathbf{x}_i, \mathbf{p}_i) x_{ij} \right) \\ \text{subject to} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_t, \ \forall i \in \mathcal{I} \\ & \sum_{i=1}^{N_u} x_{ij} \leq 1, \ \forall j \in \mathcal{J} \\ & \sum_{i=1}^{N_u} \sum_{j=1}^{N_t} x_{ij} = \min\{N_u L_t, N_t\} \\ & x_{ij} \in \{0,1\}, \ \forall (i,j) \in \mathcal{I} \times \mathcal{J} \end{aligned} \tag{1}$$

where $x_{ij} = 1$ if agent i is assigned to task j, and $\mathbf{x}_i \in \{0,1\}^{N_t}$ is a vector whose j-th 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 a sequence of tasks in the order of agent i performs; its k-th element is j if agent i conducts j at the k-th position 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* $c_{ij}(\mathbf{x}_i, \mathbf{p}_i)$ is assumed to be *nonnegative*. The score function 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 assignment for unmanned vehicles with mobility, it 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 above formulation is when $L_t = 1$ and $c_{ij}(\mathbf{x}_i, \mathbf{p}_i) \equiv c_{ij}$ without relying on $\mathbf{x}_i$ and $\mathbf{p}_i$; in this paper, this special case will be called *single assignment* in contrast to the general formulation in (1), which will be called *multi-assignment*. 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 work will first present an algorithm for the single assignment case in section III to provide conceptual insight to the consensus-based auction idea, and will then extend it to the multi assignment case in section IV with a more detailed algorithmic description.

II.B. Auction Algorithms

One of the key concepts this work is based on is the auction method for assignment problems. The auction algorithm was first proposed in Bertsekas[25] 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[25], 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 $j^\star \triangleq \operatorname{argmax}_j (a_{ij} - p_j)$. If task $j^\star$ has already been assigned to another agent, the two agents swap tasks. Once this is done, the price of task $j^\star$ is increased such that the value $c_{ij^\star}$ 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[30,33,36] to determine the winner based on the highest bids $i^\star = \operatorname{argmax}_i c_{ij}$. Other decentralized auction algorithms have been developed that remove the auctioneer in place of a different conflict resolution approaches, and allow tasks to be bid on asynchronously[49,50]. The decentralized auction approach developed herein uses a consensus algorithm for conflict resolution without need of any auctioneer.

II.C. Consensus Algorithms

For decentralized systems, cooperating agents often require a globally consistent situational awareness[18]. In a dynamic environment with sensor noise and varying network topologies, maintaining a 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[19]. 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[20–22].

In this paper, the consensus idea is used to converge on the assignment value rather than the situational awareness. 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 consensus-based auction algorithm (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, it will be shown that the CBAA can exploit the network flexibility and convergence properties of decentralized consensus algorithms, as well as the robustness and computational efficiency of the auction algorithms.

III.A. Phase 1: The 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, and $\mathbf{h}_i \in \{0,1\}^{N_t}$ be the availability vector whose j-th entry is 1 if task j is available to agent i. Two vectors of length N_t that each agent will store and update throughout the assignment process will also be 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 if not. 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 and the capability matrix, the list of valid tasks $\mathbf{h}_i$ can be generated using

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

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 enables 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. If a tie occurs in finding J_i, an agent can select one of them either randomly or lexicographically based upon the task identifier. 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.

III.B. Phase 2: The 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 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. It is assumed that 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 1 when agent i's t-th iteration corresponds to τ

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

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**

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).

If ties occur in determining z_{i,J_i}, they cannot be resolved by random selection, since tie-breaking should be conducted coherently over the fleet. Two possible ways of breaking this tie are suggested: 1) intentionally inserting a small random number to the bid, or 2) tagging the transmission packet with the agent's identification number that indicates who sent the corresponding element y_{ij} values and breaking the tie with it.

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

IV. Consensus-Based *Bundle* Algorithm

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[41–44] treated each assignment combination (bundle) as a single item for bidding which led to complicated winner selection methods. In this section, the CBAA is extended to the multi-assignment problem by presenting the consensus-based bundle algorithm (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.

IV.A. Phase 1: Bundle Construction

The first phase of the CBBA algorithm is the bundle construction process. In contrast to the previous bundle algorithms[41–44], 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 others. The tasks are added into the bundle in the following way.

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

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, 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

$$c_{ij}(\mathbf{b}_i) = \begin{cases} \max_{n \leq |\mathbf{p}_i|+1} S_i^{\mathbf{p}_i \oplus_n \{j\}} - S_i^{\mathbf{p}_i}, & \text{if } j \notin \mathbf{p}_i \\ 0, & \text{if } j \in \mathbf{p}_i \end{cases} \tag{3}$$

where $|\cdot|$ denotes the cardinality of the list, and $\oplus_n$ denotes the operation that inserts the second list right after the n-th 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 of adding the second list at the end of the first one. In other words, the CBBA scoring scheme inserts a new task at 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 does not provide any 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 is recursively updated as

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

where $J_i = \operatorname{argmax}_j c_{ij}(\mathbf{b}_i) \cdot h_{ij}$ and $n_{i,J_i} = \operatorname{argmax}_n S_i^{\mathbf{p}_i \oplus_n \{J_i\}}$ with $h_{ij} = \mathbb{I}(c_{ij} \geq y_{ij})$. The above recursion continues until either $|\mathbf{b}_i| = L_t$ or until $\mathbf{h}_i = \mathbf{0}$. Notice that with the above recursion, 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 the $\mathbf{x}_i$ used in CBAA and the $\mathbf{z}_i$ in CBBA. In CBBA, each agent needs information about not only whether or not it is outbid on the task it selects but also who is assigned to each task; this enables better assignments based on more sophisticated conflict resolution rules. These conflict resolution rules are discussed in detail in the following section.

IV.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 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, an 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 max 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 will need to be enhanced in order to ensure that these updates are accurate.

In the multi-assignment consensus stage, three vectors will be communicated for consensus. Two 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

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

where τ_r is the message reception time.

When agent i receives a message from another agent k, the $\mathbf{z}_i$ 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 1 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 1, each agent checks to see if any of the updated tasks were in their bundle, and if so, those tasks, along with all of the tasks that were added to the bundle after them, are released:

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

where b_{in} denotes the n-th entry of bundle $\mathbf{b}_i$, and $\bar{n}_i = \min n \in \{n | z_{i,b_{in}} \neq i\}$. It should be noted 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 from the task list.

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 $\mathbf{s}$ 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 $\mathbf{y}$ information is valid if it is larger than the local information regardless of the agent belief on when it is sent.

IV.C. Scoring Scheme

IV.C.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,

$$c_{ij}(\mathbf{b}_i) \geq c_{ij}(\mathbf{b}_i \oplus_{\text{end}} \mathbf{b}), \quad \forall\ \mathbf{b}_i, \mathbf{b} \in (\mathcal{J} \cup \{\emptyset\})^{L_t} \tag{7}$$

where $\emptyset$ denotes an empty task. This relation is similar to the notion of submodularity[51] for a set function except that the bundle is an ordered list rather than an unordered set; this work will refer to this condition as *diminishing marginal gain* (DMG), and satisfaction of this condition as "being DMG" in the later part.

Table 1. Action rule for agent i based on communication with agent k regarding task j

Sender's (agent k's) z_{kj}	Receiver's (agent i's) z_{ij}	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

It is true that not all the scoring functions for the multi-task assignment satisfy the above assumption. For instance, a scoring scheme with DMG cannot model some synergism by multiple selections. However, in the search and exploration problems for unmanned vehicles, many reward functions are DMG. In an exploration mission for robotic vehicles, discovery of one feature may provide knowledge about the other features' locations; thus, the marginal reward of finding other feature decreases. In a time-sensitive target assignment problem, the time-discounted reward for a target decreases if the combat UAV chooses to visit another location/task first, thereby taking a longer path to the target.

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

$$y_{i,b_{in}} \geq y_{i,b_{im}}, \text{ if } n \leq m. \tag{8}$$

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

$$y_{i,b_{in}} = \max_j c_{ij}(\mathbf{b}_i^{1:n-1}) \geq \max_j c_{ij}(\mathbf{b}_i^{1:n-1} \oplus_{\text{end}} \mathbf{b}_i^{n:m-1}) \tag{9}$$

with $\mathbf{b}_i^{k:l} \triangleq \{b_{ik}, \ldots, b_{il}\}$. In other words, the y value for the task located earlier in the bundle is never smaller than that for the task located later in the bundle.

IV.C.2. Time-Discounted Reward

In this work, the following scoring function representing the time-discounted reward will be considered with specific emphasis:

$$S_i^{\mathbf{p}_i} = \sum \lambda^{\tau_i^j(\mathbf{p}_i)} \bar{c}_j \tag{10}$$

where $\tau_i^j(\mathbf{p}_i)$ 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 search scenario in which uncertainty growth with time causes degradation of the expected reward for visiting a

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**

certain location, or planning of service routes in which satisfaction of client diminishes with time. Since the triangular inequality should satisfy for the actual distance between task locations,

$$\tau_i^j(\mathbf{p}_i \oplus_n \{k\}) \geq \tau_i^j(\mathbf{p}_i),\ \forall n, \forall k. \tag{11}$$

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

V.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 results in

$$c_{ij}^{(n+1)} = \begin{cases} c_{ij}^{(n)}, & \text{if } i \neq i_n^\star,\ j \neq j_n^\star \\ 0, & \text{if } j = j_n^\star \\ \alpha_{ij}^{(n)} c_{ij}^{(n)}, & \text{if } i = i_n^\star,\ j \neq j_n^\star \end{cases} \tag{12}$$

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 becomes zero (line 12). Thus for $\eta_i \leq L_t$ the score $c_{ij}^{(n)}$ will be *monotonically decreasing* with respect to n; namely,

$$c_{ij}^{(n)} \geq c_{ij}^{(m)},\ \text{if } n \leq m. \tag{13}$$

Also, by definition of DMG,

$$c_{ij}^{(n)} = c_{ij}(\mathbf{b}_i^{(n-1)}) \geq c_{ij}(\mathbf{b}_i^{(n-1)} \oplus_{\text{end}} \mathbf{b}),\ \forall \mathbf{b}, \tag{14}$$

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 n-th step, $(i_n^\star, j_n^\star)$ in line 5, gives the largest score given selections up to the $(n-1)$-th step,

$$c_{i_n^\star, j_n^\star}^{(n)} \geq c_{ij}^{(n)}, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J}. \tag{15}$$

Therefore, notice that from (13) and (15),

$$c_{i_n^\star, j_n^\star}^{(n)} \geq c_{i_m^\star, j_m^\star}^{(n)} \geq c_{i_m^\star, j_m^\star}^{(m)} \geq c_{ij}^{(m)}, \ \forall (i,j) \in \mathcal{I} \times \mathcal{J}, \ \text{if } n \leq m. \tag{16}$$

Namely, the best score at the n-th step is greater than, or equal to, any score value showing up in the later steps.

V.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:

$$D = \max_{i,k} d_{ik}. \tag{17}$$

If the conflict resolution is assumed to be *synchronized*, i.e., every agent's second phase in the t-th 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:

$$T_C \triangleq \min t \in \mathcal{T} \tag{18}$$

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

$$\begin{aligned} \mathcal{T} = \Big\{ t \in \mathbb{Z}_+ \ \big| \forall s \geq t : x_{ij}(s) = x_{ij}(t), \textstyle\sum_{i=1}^{N_u} x_{ij}(s) = 1, \\ \textstyle\sum_{j=1}^{N_t} x_{ij}(s) \leq L_t, \ \sum_{j=1}^{N_t} \sum_{i=1}^{N_u} x_{ij}(s) = N_{\min} \Big\} \end{aligned} \tag{19}$$

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,

$$z_{i,j_k^\star}(t) = i_k^\star, \ y_{i,j_k^\star}(t) = c_{i_k^\star, j_k^\star}^{(k)}, \ \forall i \in \mathcal{I}, \ \forall k \leq n, \tag{20}$$

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 are true:

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 n-th SGA step, $\mathbf{b}_i^{(n)}$:

$$\mathbf{b}_i^{1:L_i^{(n)}} = \mathbf{b}_i^{(n)}. \tag{21}$$

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

$$y_{i_{n+1}^\star, j_{n+1}^\star}(t) = c_{i_{n+1}^\star, j_{n+1}^\star}^{(n)}, \tag{22}$$

and this value satisfies

$$y_{i_{n+1}^\star, j_{n+1}^\star}(t) \geq y_{ij}(t), \ \forall (i,j) \in \mathcal{I}_{n+1} \times \mathcal{J}_{n+1}. \tag{23}$$

3. Entries in (20) do not change over time; or,

$$z_{i,j_k^\star}(s) = z_{i,j_k^\star}(t), \quad y_{i,j_k^\star}(s) = y_{i,j_k^\star}(t), \tag{24}$$

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

4. The bid agent $i^\star_{n+1}$ places on task $j^\star_{n+1}$ will remain the same and it will not be outbid:

$$y_{i^\star_{n+1},j^\star_{n+1}}(s) = y_{i^\star_{n+1},j^\star_{n+1}}(t) \geq y_{i,j^\star_{n+1}}(s). \tag{25}$$

for all $s \geq t$ 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,

$$y_{i,j^\star_{n+1}}(t+D) = y_{i^\star_{n+1},j^\star_{n+1}}(t), \; z_{i,j^\star_{n+1}}(t+D) = i^\star_{n+1}. \tag{26}$$

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

Proof. See Appendix A. □

Lemma 2. In the CBBA process with synchronous conflict resolution over a static network of diameter D for the case every agent's scoring scheme satisfies diminishing marginal gain condition, every agent has agreed on the first n SGA assignments by iteration nD; in other words,

$$z_{i,j_k^\star}(nD) = i_k^\star, \qquad \forall i \in \mathcal{I}, \; \forall k \leq n, \text{and} \tag{27}$$

$$y_{i,j_k^\star}(nD) = c^{(k)}_{i_k^\star,j_k^\star}, \qquad \forall i \in \mathcal{I}, \; \forall k \leq n. \tag{28}$$

Proof. The proof is by induction. Since $\text{argmax}_{j\in\mathcal{J}} c_{i_1^\star,j}(\{\emptyset\}) = j_1^\star$, agent $i_1^\star$ places task $j_1^\star$ in the first position of its bundle in phase 1 of iteration 1. Because $c_{i_1^\star,j_1^\star}(\{\emptyset\}) \geq c_{ij}(\mathbf{b})$ for all $(i,j) \in \mathcal{I}\times\mathcal{J}$ for any $\mathbf{b}$, no one can place a higher bid in the later iterations. Thus, the k iterations of CBBA conflict resolution procedures leads k-hop neighbors of agent $i_1^\star$ to agree that $i_1^\star$ is the winning agent for task $j_1^\star$. Thus, after D iterations of phase 2, every agent will have agreed on the assignment $(i_1^\star, j_1^\star)$. Due to the statement 3 and 5 in Lemma 1, if $z_{i,j_k^\star}(mD) = i_k^\star$, $y_{i,j_k^\star}(mD) = c^{(k)}_{i_k^\star,j_k^\star}$ for all $k \leq m$, then $z_{i,j_k^\star}(mD+D) = i_k^\star$, $y_{i,j_k^\star}(mD+D) = c^{(k)}_{i_k^\star,j_k^\star}$ for all $k \leq m+1$. Thus, together with $(i_1^\star, j_1^\star)$ being agreed at D, after nD iterations, every agent will have agreed on the assignments $(i_k^\star, j_k^\star)$ for all $k \leq n$. □

Theorem 1. 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.

$$\begin{aligned} z_{i,j_k^\star} &= i_k^\star, \; \forall k \leq N_{\min} \triangleq \min\{L_t N_u, N_t\}, \\ y_{i,j_k^\star} &= c^{(k)}_{i_k^\star,j_k^\star}, \; \forall k \leq N_{\min}. \end{aligned} \tag{29}$$

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 the statement 3, these assignments will not change in the later iterations. Thus, at latest at iteration $N_{\min}D$, the CBBA process converges and the converged solutions are equivalent to SGA solution. □

Note that in many cases CBBA converges much earlier than $N_{\min}D$ iterations, because the maximum distance from some $i_k^\star$ 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.

V.C. Dynamic Network and Asynchronous Conflict Resolution

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

$$\mathbb{W}(t) = \mathbb{G}(t) \cup \mathbb{G}(t+1) \cup \cdots \cup \mathbb{G}(t+\rho-1) \tag{30}$$

is fully connected $\forall t$[18]. In this case, the convergence time will then be upper-bounded by $\rho N_{\min}$, since any information about confliction 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 asynchronous conflict resolution is allowed.

V.D. Inconsistent Information

It is typical that each agent's scoring scheme will be based on its own understanding of the environment (commonly known as the situational awareness). For instance, the time-discounted reward in (10), the score depends on the target and agent locations; so with a different estimates of either, the resulting scores used by the agents in CBBA will differ. Since these scores will 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 situational awareness 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. This is unlike algorithms like implicit coordination[13] or the ETSP auction algorithm[49,50] in which each agent must have the same information to guarantee convergence. The CBBA on the other hand, does not require any level of agreement for convergence, although inconsistent information might still cause actual performance degradation.

V.E. Dynamic Scoring

Although no explicit dependency on the time of the scoring scheme, CBBA can take into account the time-varying aspect of the score functions. In this case, the only change in the algorithm description is the score value c_{ij} (in Algorithm 1 and Algorithm 3) has a time argument τ as well. Even in the case the scoring scheme is time-varying, the CBBA process can be treated as max consensus procedure on $\mathbf{y}_i$ vector. Define $\bar{\mathbf{y}}$ as the vector that every agent's winning bid list supposedly converges to; CBBA converges if it is ensured $\bar{y}_j$ is never outbid once it is first introduced in the CBBA process, which is the case if the scoring scheme satisfies the following two properties:

$$\begin{aligned} &c_{ij}(\tau_1) \geq c_{ij}(\tau_2), \quad \forall \tau_1 \leq \tau_2, \\ &c_{ij}(\tau_1) \geq c_{kj}(\tau_1) \Rightarrow c_{ij}(\tau_2) \geq c_{kj}(\tau_2), \quad \forall \tau_1 \leq \tau_2. \end{aligned} \tag{31}$$

In other words, if the scoring scheme is a monotonically decreasing function of time, and if it maintains the superiority between agents on a task, then no one can place a bid better than $\bar{y}_{i_n^\star}$ introduced at time τ_1 on task $j_n^\star$. Thus, CBBA can produce a solution to a more general class of problems than described in (1). For instance, dependency of the reward on the agent motion can be incorporated into the assignment process. For many missions involving autonomous operation, it is realistic to assume that the task values will decrease with time as missions are often designed to minimize time, fuel or exposure to threats. Similar scoring systems have also been developed in Moore and Passino[52] and shown to produce better assignments in limited communication networks.

The above argument enables convergence of CBBA process with a scoring scheme that does not satisfy the diminishing marginal gain property. Notice that what is indeed important for convergence is whether or not a best greedy bid is outbid after introduced in the process; the diminishing marginal gain property is one such case that ensures it will never be outbid. Thus, if the original scoring scheme does not provide

diminishing marginal gain, it is conceivable to modify the scoring scheme to satisfy the conditions in (31) as follows.

Remark 1. Even in the case that a scoring function does not represent diminishing marginal gain, the following dynamic modification of the scoring scheme ensures a monotonically decreasing property of a bid over time will lead to convergence

$$c_{ij}(t) = \min\left\{c_{ij}(t), c_{ij}(t-1)\right\}. \tag{32}$$

Thus, in this case each agent keeps track of one more vector $\mathbf{c}_i(t) \in \mathbb{R}^{N_t}$ and carries out (32) between line 7 and line 8 of the phase 1 of CBBA (Algorithm 3).

VI. Performance

The previous section proved that CBBA creates the same solution as the centralized sequential greedy selection procedure in case scoring schemes are DMG. This enables performance analysis of CBBA solutions by instead investigation the properties of SGA solutions. While the actual performance of CBBA (and CBAA) is problem dependent, the worst-case performance bound that considers the guaranteed performance for any possible DMG scoring schemes can be analytically derived. In addition, the average performance of CBAA can be analyzed by assuming some prior knowledge of probabilistic distribution of score values.

VI.A. Minimum Performance Guarantee

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

- **SOPT**: the optimal objective value of the single assignment problem for a given non-negative scoring scheme.
- **CBAA**: the objective value provided by CBAA for the single assignment problem for a given non-negative scoring scheme.
- **MOPT**: the optimal objective value of the multi-assignment problem for a given non-negative DMG scoring scheme.
- **CBBA**: the objective value provided by CBBA for the multi-assignment problem for a given non-negative 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:

Proposition 1. *(CBAA Performance Bound)* Assuming the agents have accurate knowledge the of situational awareness, CBAA guarantees 50% optimality. In other words,

$$\mathbf{SOPT} \leq 2 \cdot \mathbf{CBAA}. \tag{33}$$

Proof. Since the CBAA 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:

$$i_k^\star = k, \quad j_k^\star = k, \quad \forall k \leq N_{\min}. \tag{34}$$

In other words, for now call 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 (16) for SGA,

$$c_{ii} \geq c_{jj}, \text{ if } i < j, \tag{35}$$

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

$$\mathbf{CBAA} = \sum_{i=1}^{N_{\min}} c_{ii}. \tag{36}$$

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

$$\begin{aligned} c_{ii} &\geq c_{ij}, \ \forall i, \ \forall j > i \\ c_{ii} &\geq c_{ji}, \ \forall i, \ \forall j > i. \end{aligned} \tag{37}$$

Consider the case the greedy selection is the farthest from the optimal solution; in other words, think of the case where variations of assignment could cause the largest improvement in the objective value while still satisfying the conditions in (37). 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 (37) is satisfied, the new overall score $c_{ij} + c_{ji}$ is upperbounded by

$$c_{ij} + c_{ji} \leq c_{ii} + c_{ii} = 2c_{ii}, \tag{38}$$

where the upperbound is attained if

$$c_{ij} = c_{ji} = c_{ii}. \tag{39}$$

Thus, if (39) is satisfied, agents i and j can increase their overall score the most by swapping their tasks. Now suppose that the similar condition to (39) holds for all pairs of agents:

$$\begin{aligned} c_{ij} &= c_{ii}, \quad \forall i, \ \forall j > i \\ c_{ji} &= c_{ii}, \quad \forall i, \ \forall j > i, \end{aligned} \tag{40}$$

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

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

$$J_i^* = \begin{cases} N_{\min} - i + 1, & \text{if } i \in \{1, \ldots, N_{\min}\} \\ \emptyset, & \text{otherwise,} \end{cases} \tag{41}$$

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 to 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 (41) ensures one agent to be 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

$$\begin{aligned} \mathbf{SOPT} &= \sum_{i=1}^{\lceil N_{\min}/2 \rceil} c_{ii} + \sum_{i=\lceil N_{\min}/2 \rceil + 1}^{N_{\min}} c_{(N_{\min}-i+1),(N_{\min}-i+1)} \\ &= 2 \times \sum_{i=1}^{\lfloor N_{\min}/2 \rfloor} c_{ii} + \sum_{i=\lfloor N_{\min}/2 \rfloor + 1}^{\lceil N_{\min}/2 \rceil} c_{ii} \leq 2 \times \sum_{i=1}^{N_{\min}} c_{ii} = 2 \cdot \mathbf{CBAA}. \end{aligned} \tag{42}$$

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

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

Proposition 2. *(CBBA Performance Bound)* Assuming the agents have accurate knowledge of the situational awareness, CBBA guarantees 50% optimality for the multiple assignment problem with DMG scoring schemes:

$$\mathbf{MOPT} \leq 2 \cdot \mathbf{CBBA}. \tag{43}$$

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

Remark 2. The 50% performance lower bounds are the tightest bounds for CBAA and CBBA, as they can be achieved in the following examples:
1) (CBAA Example) Consider the score matrix

$$C = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \tag{44}$$

whose (i, j)-the element is c_{ij}, and suppose that CBAA breaks a tie such that agent 1 takes the task 1. The CBAA solution assigns agent i to task i for $i = 1, 2$, which results in objective value of 1. But, if agent i is assigned to $3 - i$ for $i = 1, 2$, then the objective value becomes 2 and this solution is optimal because it is the better between two feasible solutions. Namely, $\mathbf{SOPT} = 2 \cdot \mathbf{CBAA}$ for this example.
2) (CBBA Example) Consider the case where the single assignment score matrix is

$$C = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \tag{45}$$

and the first agent's score vector after selecting task 1 is

$$\mathbf{c}_1(\{1\}) = \begin{bmatrix} 0 & 0 \end{bmatrix}. \tag{46}$$

Suppose CBBA breaks a tie such that agent 1 takes task 1 in the first selection. Then, the CBBA objective value is 1 whatever agent selects the task 2. However, if agent 2 takes task 2 in the first step, then agent 1 will take task 1 in the second step; this process corresponds to the optimal solution with objective value of 2. Hence, $\mathbf{MOPT} = 2 \cdot \mathbf{CBBA}$ in this example.

VI.B. Expected Performance for CBAA

The previous two propositions showed that both CBAA and CBBA provides at least 50% as best objective value as the optimal solution and that this lower bound is the tightest bound. This section addresses a quantity of more practical interest – the expected performance, where the expectation is taken over the possible score scheme from a known probability distribution. Specifically, it will be demonstrated that for the single assignment problem where each element of the score matrix is uniformly i.i.d, a closed form expression of the upper bound of the expected optimality gap can be derived.

First, note that if each element of the scoring matrix is distributed i.i.d,

$$\mathbb{E}[\mathbf{Sequence}] \leq \mathbb{E}[\mathbf{CBAA}] \leq \mathbb{E}[\mathbf{OPT}] \leq \mathbf{E}[\mathbf{InfOpt}].$$

where **Sequence** and **InfOpt** are defined as:

- **Sequence**: the objective value based on the strategy that with pre-defined agent precedence, the lower-indexed agent greedily selects a task and then the next highest indexed agent greedily selects while taking conflicts into account.
- **InfOpt**: the objective value based on the strategy that each agent selects greedily without considering conflict resolution.

In case every score value is i.i.d, **Sequence** should not depend on specific precedence order; thus, it represents the performance of a randomly-ordered greedy selection process, which has no reason to be better than CBAA. Also, since **InfOpt** considers a relaxation of the original problem, it should provide an upper bound of the optimal solution. Thus, an upperbound of the optimality gap can be obtained as follows:

$$\mathcal{E} \triangleq 1 - \frac{\mathbb{E}[\mathbf{CBAA}]}{\mathbb{E}[\mathbf{OPT}]} \leq 1 - \frac{\mathbb{E}[\mathbf{Sequence}]}{\mathbb{E}[\mathbf{InfOpt}]} \triangleq \bar{\mathcal{E}}. \tag{47}$$

Thus, if $\mathbb{E}[\mathbf{InfOpt}]$ and $\mathbb{E}[\mathbf{Sequence}]$ can be calculated easily compared to $\mathbb{E}[\mathbf{OPT}]$ and $\mathbb{E}[\mathbf{CBAA}]$, the upperbound $\bar{\mathcal{E}}$ can provide useful indication of how close the CBBA solution is to the optimum.

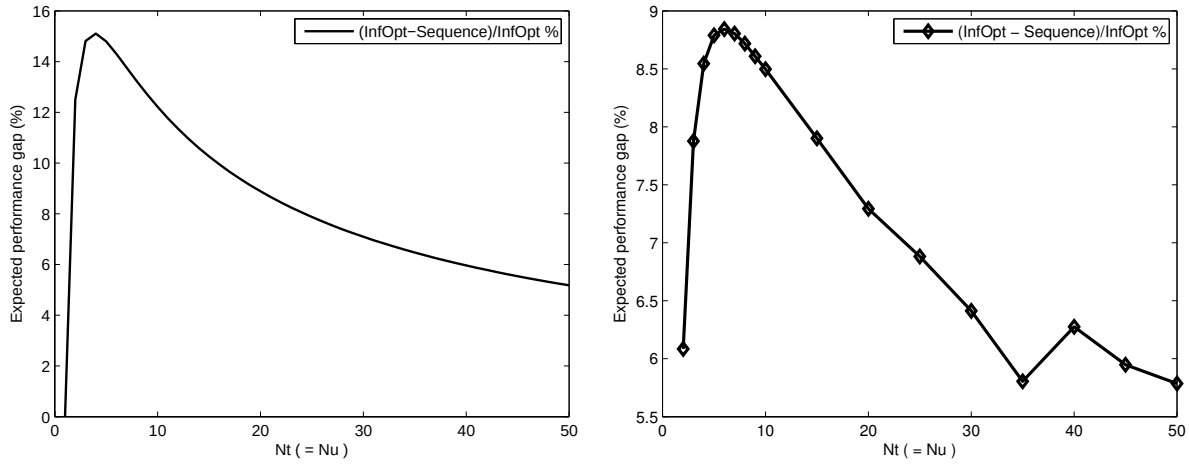

Figure 1. Upperbound on the optimality gap (a) for the uniformly distributed c_{ij}, (b) for the case with uniformly distributed agents and targets in a two-dimensional space.

VI.B.1. Uniform score matrix

In the case $c_{ij} \sim \mathcal{U}[0, c_{\max}]$, $\forall(i,j)$, $\mathbb{E}[\textbf{InfOpt}]$ and $\mathbb{E}[\textbf{Sequence}]$ can be obtained in a closed form by using the order statistics. Regarding the **InfOpt** case, each agent selects the largest entry from N_t realizations of i.i.d samples. Thus, the distribution of each agent's selection corresponds to N_t-th order statistics of $\mathcal{U}[0, c_{\max}]$. It is known that the k-th order statistics (k-th smallest one) from n samples taken from $\mathcal{U}[0,1]$ has the Beta distribution[53] $U_{(k)} \sim \text{Beta}(k, n+1-k)$ and its mean is $\mathbb{E}[U_{(k)}] = \frac{k}{n+1}$. Then, the expected performance is simply the sum of each agent's expected performance:

$$\mathbb{E}[\textbf{InfOpt}] = N_{\min}\mathbb{E}[c_{\max}U_{(N_t)}] = N_{\min}c_{\max}\left[1 - \tfrac{1}{N_t+1}\right] \tag{48}$$

where $N_{\min} = \min\{N_u, N_t\}$. On the other hand, in the **Sequence** case, the expected performance of i-th agent is the expected value of the largest order statistics out of $(N_t - i + 1)$ samples. Therefore,

$$\begin{aligned}\mathbb{E}[\textbf{Sequence}] &= \textstyle\sum_{i=1}^{N_{\min}} \mathbb{E}[c_{\max}U_{(N_t-i+1)}] \\ &= N_{\min}c_{\max} - c_{\max}\textstyle\sum_{i=1}^{N_{\min}}(N_t - i + 2)^{-1}.\end{aligned} \tag{49}$$

Thus, the expected optimality gap is bounded above as:

$$\mathcal{E} \le -\tfrac{1}{N_t} + \tfrac{1}{N_{\min}}\left(1 + \tfrac{1}{N_t}\right)\textstyle\sum_{i=1}^{N_{\min}}(N_t - i + 2)^{-1} \triangleq \bar{\mathcal{E}}. \tag{50}$$

Figure 1(a) plots $\bar{\mathcal{E}}$ when $N_t = N_u = N_{\min}$ for different N_t values. Note first that the analytic upperbound indicates that the average performance of CBAA is much better than the 50% worst-case performance and becomes closer to the optimum as the problem size increases. The expression in (50) leads to

$$0 \le \lim_{N_t\to\infty} \mathcal{E} \le \lim_{N_t\to\infty} \bar{\mathcal{E}} = 0, \tag{51}$$

which is equivalent to

$$\lim_{N_t\to\infty} \mathcal{E} = 0. \tag{52}$$

VI.B.2. Uniformly distributed agents and targets

Consider the situation where N_u agents and N_t targets are uniformly distributed over a two-dimensional space $[0, L] \times [0, L]$, and the goal is to assign agents to targets to maximize the sum of each agent's time-discounted reward. In this case, the distribution of each element in the scoring matrix is i.i.d, thus, a similar analysis as the previous section can be done to derive the performance bound of the CBAA in this setup. The first step is to derive the probability density of each entry of the scoring matrix: $c_{ij} = c_0 \exp(-r_{ij}/r_0)$,

where c_0 and r_0 are constants. If the probability density function (pdf) of r_{ij} is known, the pdf of c_{ij} can be expressed as

$$f_C(c) = \begin{cases} \frac{r_0}{c} f_R\left(-r_0 \log(c/c_0)\right), & \text{if } c \in [c_0 e^{-\sqrt{2}L/r_0}, c_0], \\ 0, & \text{otherwise,} \end{cases} \tag{53}$$

where $f_R(r)$ is the pdf of the distance between an agent and target (indices are omitted to avoid confusion). Since the positions of an agent and target have uniform distribution, the coordinate difference of x and y have a triangular distribution:

$$f_X(x) = \begin{cases} \frac{1}{L}\left(1 + \frac{x}{L}\right), & \text{if } x \in [-L, 0] \\ \frac{1}{L}\left(1 - \frac{x}{L}\right), & \text{if } x \in [0, L] \\ 0, & \text{otherwise,} \end{cases} \tag{54}$$

and $f_Y(y)$ has the same form. The pdf of r is related to the pdfs of x and y as follows:

$$f_R(r) = \int_0^{2\pi} f_X(r\cos\theta) f_Y(r\sin\theta) d\theta. \tag{55}$$

It can be shown that $f_R(r)$ can be derived as a closed form:

$$f_R(r) = \begin{cases} \frac{r}{L^2}\left[2\pi - \frac{8r}{L} + \frac{2r^2}{L^2}\right], & \text{if } r \in [0, L] \\ \frac{r}{L^2}\left[4(\sin^{-1}\frac{L}{r} - \cos^{-1}\frac{L}{r}) + 8(\sqrt{\frac{r^2}{L^2} - 1} - 1) + (4 - 2\frac{r^2}{L^2})\right], & \text{if } r \in [L, \sqrt{2}L] \\ 0, & \text{otherwise.} \end{cases} \tag{56}$$

Figure 2 depicts $f_C(c)$ with $c_0 = 1$ for five different values of r_0: $L/4, L/2, L, 2L, 4L$. Given $f_C(c)$, the k-th order statistics out of i.i.d samples of size N_t is represented as

$$f_{C(k)}(c) = N_t \binom{N_t - 1}{k - 1} F_C(c)^{k-1} (1 - F_C(c))^{N_t - k} f_C(c). \tag{57}$$

Figure 1(b) illustrates the upper bound of the optimality gap, $1 - \mathbb{E}[\textbf{Sequence}]/\mathbb{E}[\textbf{InfOpt}]$, in the case that $r_0 = L$ for different values of $n = N_t = N_u$. It can be seen that the performance of the CBAA provides less than a 9% optimality gap, and that the gap becomes smaller as the problem size increases.

Monte-Carlo simulations in section will verify that the actual performance of CBBA is better than the analytical upperbounds derived in this section.

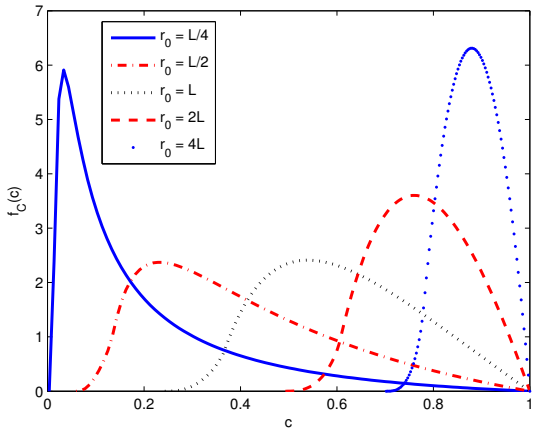

Figure 2. Probability density for the score value for the case with uniformly deployed agents and targets

VII. Numerical Results

VII.A. Assignment with Time-discounted Rewards

This section first compares the performance of CBBA (and CBAA) with the optimal solution using Monte-Carlo simulations, where the time discounted reward in (10) defines the scoring function. The agents and tasks are randomly placed on a $W \times W$ two-dimensional space with uniform distribution, and every task incurs the same fixed score before discounting. The discount rate is $\lambda = 0.95$, while $W = 2$ km and vehicle speed is 40 m/s. The network topology, networks are created by generating a random spanning tree[54], and then adding varying amounts of random links to the network. This created random network diameters ranging from $D = 1$ to $D = N_u - 1$. The optimal solution is obtained by implicit coordination algorithm[7].

Figure 3(a) depicts the average percent optimality gap $(1 - \textbf{CBBA}/\textbf{MOPT}) \times 100$ with varying problem sizes in terms of number of agents N_u and tasks N_t, and the maximum number of tasks an agent can take L_t. This simulation sets $N_u = N_t$ since the performance gap is even smaller for other cases. The maximum

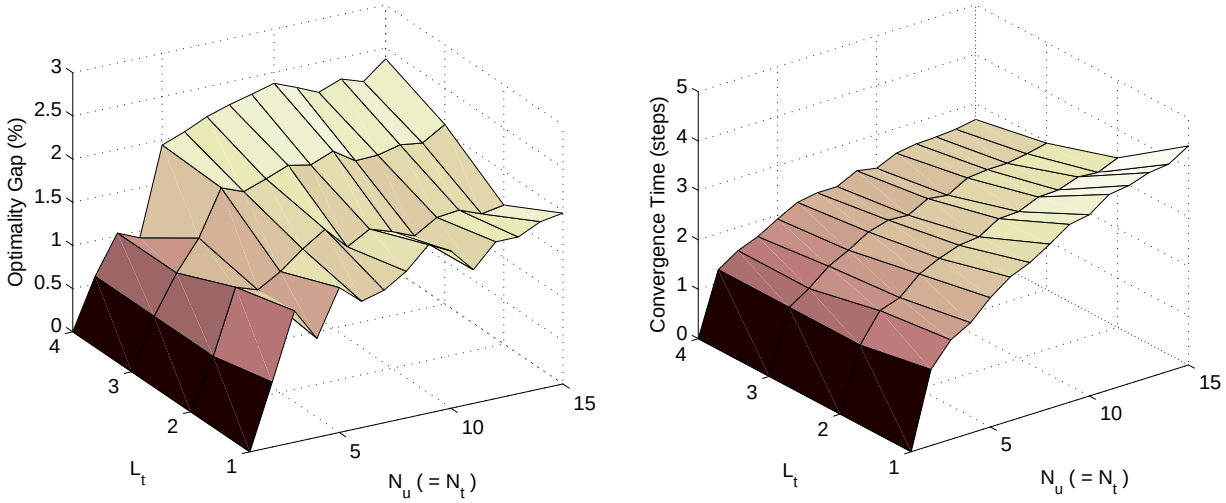

Figure 3. (a) Optimality gap of CBBA solution, (b) Convergence time steps of CBBA.

value of $N_u = N_t$ and L_t are determined by the tractability of finding the optimal solution. It should be first noted that CBBA produces solutions very close to the optimum, specifically, within 3% optimality gap in this simulation.

In the same setting, the number of convergence time steps are illustrated in Figure 3(b). The convergence time step means the number of iterations for convergence in synchronized implementation of CBBA, and it can be regarded as a measure of communication cost since it also represents the number of communication rounds for convergence. It can be found in Figure 3(b) that the algorithm converges faster than $\mathcal{O}(N_u)$ time steps, and also that increase of L_t accelerates the convergence. For large L_t, more tasks tends to be assigned per iteration, and the final solution tends to be found within a fewer time steps.

Similar simulations were performed to confirm the robust convergence of CBBA process in the presence of inconsistencies in situational awareness over the fleet, and the results showed that the convergence time is not significantly impacted by mismatches in SA[55]. Also, comparisons with an existing centralized sequential auction algorithm[32] showed that CBBA provides much faster convergence[55].

VII.B. Assignment with Agent Motion

As discussed in section V.E, CBBA can handle a time dependency of the scoring scheme. One important source of variation of the score in time is vehicle motions. This section verifies the capability and performance of the proposed method in this dynamic assignment by comparing with an existing decentralized assignment algorithm tailored to handle vehicle motion, the ETSP algorithm[49,50]. The ETSP algorithm addresses a single assignment problem with vehicle motion, removes the auctioneer similarly to CBAA, and has a novel message passing system to diminish the number of conflicts. Since ETSP assumes perfect knowledge, it should be noted that CBAA has superior robustness against inconsistent information. Aside from this robustness benefit, this section compares the performance of CBAA and ETSP, in particular, when each agent has limited communication range and thus the topology of communication network changes over time with agents' motions. In addition, the greedy-based auction algorithm (GBAA) is also considered in this comparison. In GBAA, agents greedily select tasks and resolve conflicts with their direct neighbors only.

Tasks and agents are placed in a same manner as previous numerical simulations. Tasks are held stationary while agents are able to move with a predefined constant speed. The task positions are known to the agents *a priori* and the objective is to *minimize* the sum of each agent's time of arrival. $N_u = 5, 20$ are considered with the same number of tasks ($N_t = N_u$), and Monte-Carlo simulations are carried out with varying the communication range of the vehicles. Figures 4(a) and 4(b) show the average objective values – the total mission completion times – for the above mentioned three algorithms and the optimal solutions, for the cases of 5 and 20 agents, respectively. The x axis represents the normalized communication range defined as $R_{\text{comm}}/(\sqrt{2}W)$; the y axis indicates the average mission completion time based on 100 Monte-Carlo simulations for each data point. The results indicate that, in the case that the communication range

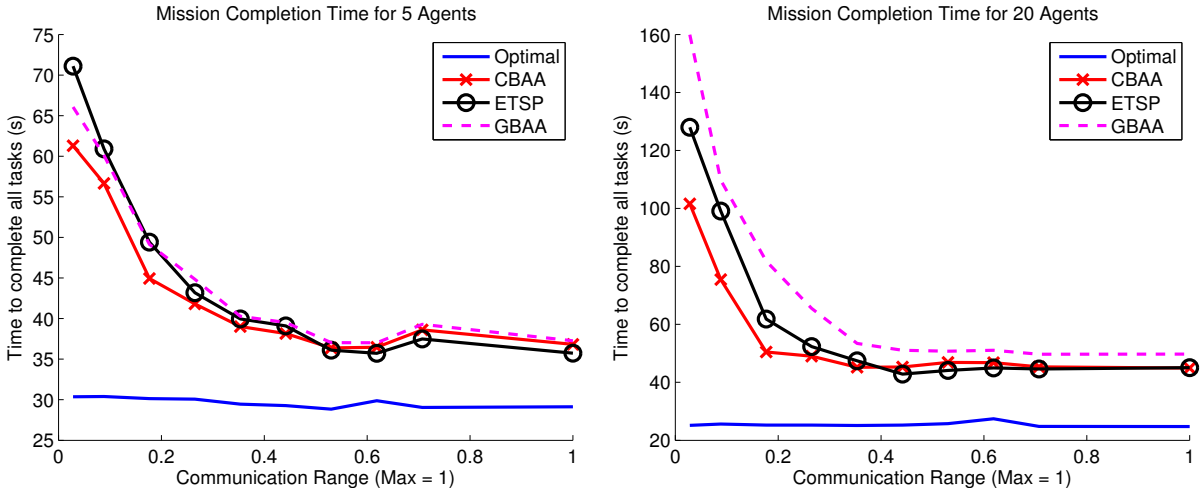

Figure 4. (a) By comparing the CBAA to the GBAA and ETSP algorithms, it is shown to have better performance as the communication range is decreased, (b) The addition of multiple agents to the fleet allows the CBAA to makes use of the extra edges for faster conflict resolution resulting in closer-to-optimal solutions than those found using the GBAA and ETSP cases.

is very short, CBAA outperforms both the ETSP and GBAA strategies, while all three are comparable to each other in high communication environments. This is because the consensus phase in CBAA enables each agent to access the global information earlier than via one-step propagation as in GBAA or than via pruning of unselected tasks as in ETSP. Furthermore, as the number of agents is increased, the deviation between CBAA and both ETSP and GBAA strategies is increased for low communication range values. This result follows because an increase in the number of agents for a given communication range effectively enhances the network connectivity, which enables CBAA to resolve conflicts more quickly. For the dynamic assignment problem with objective of the mission completion time, quick convergence can lead to better performance, because it reduces the time each agent is wandering around to find the solution.

VIII. Conclusions

This paper presented two decentralized task assignment algorithms addressing single and multi-assignment problems, respectively, that utilize the auction-based task selection procedure and the consensus-based conflict resolution process. Under some mild assumptions on the network connectivity and the scoring scheme, the proposed algorithms were proven to generate feasible solutions identical to the solutions of the centralized greedy algorithm regardless of the details of the network topology and exhibit guaranteed worst-case performance and provably good average performance. Numerical experiments using the static and dynamic assignment examples validated good performance and quick convergence of the proposed methods, and indicated better performance than an existing dynamic assignment algorithm under limited communication.

Acknowledgment

This work is funded in part by AFOSR STTR # FA9550-06-C-0088 (with Dr. Jim Paduano at Aurora Flight Sciences) and by AFOSR # FA9550-08-1-0086. The authors thank Dr. Mehdi Alighanbari for his invaluable contribution in development of the precursors to the consensus-based auction algorithm.

References

[1] Bellingham, J., Tillerson, M., Richards, A., and How, J., "Multi-Task Allocation and Path Planning for Cooperating UAVs," *Proceedings of Conference of Cooperative Control and Optimization*, Nov. 2001.

[2] Schumacher, C., Chandler, P., and Rasmussen, S., "Task Allocation For Wide Area Search Munitions," *Proceedings of the American Control Conference*, 2002.

[3] Cassandras, C. and Li, W., "A Receding Horizon Approach for Solving Some Cooperative Control Problems,"

Proceedings of the IEEE Conference on Decision and Control, 2002.

[4] Jin, Y., Minai, A., and Polycarpou, M., "Cooperative Real-Time Search and Task Allocation in UAV Teams," *Proceedings of the IEEE Conference on Decision and Control*, 2003.

[5] Xu, L. and Ozguner, U., "Battle Management for Unmanned Aerial Vehicles," *Proceedings of the IEEE Conference on Decision and Control*, 2003.

[6] Turra, D., Pollini, L., and Innocenti, M., "Fast Unmanned Vehicles Task Allocation with Moving Targets," *Proceedings of the IEEE Conference on Decision and Control*, Dec 2004.

[7] Alighanbari, M., *Task assignment algorithms for teams of UAVs in dynamic environments*, Master's thesis, Massachusetts Institute of Technology, 2004.

[8] McLain, T. W. and Beard, R. W., "Coordination Variables, Coordination Functions, and Cooperative-Timing Missions," *Journal of Guidance, Control, and Dynamics*, Vol. 28(1), 2005, pp. 150–161.

[9] Castanon, D. and Wu, C., "Distributed Algorithms for Dynamic Reassignment," *Proceedings of the IEEE Conference of Decision and Control*, 2003.

[10] Curtis, J. and Murphey, R., "Simultaneous Area Search and Task Assignment for a Team of Cooperative Agents," *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003.

[11] Shima, T., Rasmussen, S. J., and Chandler, P., "UAV Team Decision and Control using Efficient Collaborative Estimation," *Proceedings of the American Control Conference*, 2005.

[12] Fax, J. A. and Murray, R. M., "Information Flow and Cooperative Control of Vehicle Formations," *IEEE Transactions on Automatic Control*, Vol. 49(9), 2004, pp. 1465–1476.

[13] Ren, W., Beard, R., and Kingston, D., "Multi-agent Kalman Consensus with Relative Uncertainty," *Proceedings of American Control Conference*, 2005.

[14] Olfati-Saber, R. and Murray, R. M., "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays," *IEEE Transactions on Automatic Control*, Vol. 49(9), 2004, pp. 1520–1533.

[15] Alighanbari, M. and How, J. P., "An Unbiased Kalman Consensus Algorithm," *Proceedings of the American Control Conference*, 2006.

[16] Moallemi, C. C. and Roy, B. V., "Consensus Propagation," *IEEE Transactions on Information Theory*, Vol. 52(11), 2006, pp. 4753–4766.

[17] Olshevsky, A. and Tsitsiklis, J. N., "Convergence Speed in Distributed Consensus and Averaging," *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006.

[18] Ren, W., Beard, R. W., and Atkins, E. M., "Information consensus in multivehicle control," *IEEE Control Systems Magazine*, Vol. 27(2), 2007, pp. 71–82.

[19] Beard, R. and Stepanyan, V., "Synchronization of Information in Distributed Multiple Vehicle Coordinated Control," *Proceedings of the IEEE Conference on Decision and Control*, 2003.

[20] Hatano, Y. and Mesbahi, M., "Agreement over Random Networks," *43rd IEEE Conference on Decision and Control*, 2004.

[21] Wu, C. W., "Synchronization and Convergence of Linear Dynamics in Random Directed Networks," *IEEE Transactions on Automatic Control*, Vol. 51(7), 2006, pp. 12071210.

[22] Tahbaz-Salehi, A. and Jadbabaie, A., "On Consensus Over Random Networks," *44th Annual Allerton Conference*, 2006.

[23] Alighanbari, M. and How, J. P., "Decentralized Task Assignment for Unmanned Aerial Vehicles," *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, 2005.

[24] Dionne, D. and Rabbath, C. A., "Multi-UAV Decentralized Task Allocation with Intermittent Communications: the DTC algorithm," *Proceedings of the American Control Conference*, 2007.

[25] Bertsekas, D. P., "The Auction Algorithm for Assignment and Other Network Flow Problems," Tech. rep., MIT, 1989.

[26] Dias, M. B., Zlot, R., Kalra, N., and Stentz, A., "Market-Based Multirobot Coordination: A Survey and Analysis," *Proceedings of the IEEE*, Vol. 94(7), 2006, pp. 1257–1270.

[27] Gerkey, B. and Mataric, M., "Sold!: Auction Methods for Multirobot Coordination," *IEEE Transactions on Robotics and Automation*, Vol. 18(5), 2002, pp. 758–768.

[28] Bertsekas, D. P., "Auction Algorithms," *Encyclopedia of Optimization*, Kluwer Academic Publishers, 2001.

[29] Gerkey, B. P. and Mataric, M. J., "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *International Journal of Robotics Research*, Vol. 23(9), 2004, pp. 939–954.

[30] Kwasnica, A. M., Ledyard, J. O., Porter, D., and DeMartini, C., "A New and Improved Design for Multiobject Iterative Auctions," *Management Science*, Vol. 51(3), 2005, pp. 419434.

[31] Milgrom, P., "Putting Auction Theory to Work: The Simultaneous Ascending Auction," *The Journal of Political Economy*, Vol. 108(2), 2000, pp. 245–272.

[32] Lagoudakis, M. G., Berhaultt, M., Koenigt, S., Keskinocak, P., and Kleywegt, A. J., "Simple Auctions with Performance Guarantees for Multi-Robot Task Allocation," *Proceedings of the IEEE/RSI International Conference*

on Intelligent Robots and Systems, 2004.
[33] Sariel, S. and Balch, T., "Real Time Auction Based Allocation of Tasks for Multi-Robot Exploration Problem in Dynamic Environments," *Proceedings of the AIAA Workshop on "Integrating Planning Into Scheduling"*, 2005.
[34] Ahmed, A., Patel, A., Brown, T., Ham, M., Jang, M., and Agha, G., "Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism," *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2005.
[35] Atkinson, M. L., "Results Analysis of Using Free Market Auctions to Distribute Control of UAVs," *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004.
[36] Lemaire, T., Alami, R., and Lacroix, S., "A Distributed Task Allocation Scheme in Multi-UAV Context," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.
[37] Walsh, W. and Wellman, M., "A market protocol for decentralized task allocation," *Proceedings of International Conference on Multi Agent Systems*, 1998.
[38] Hoeing, M., Dasgupta, P., Petrov, P., and OHara, S., "Auction-based Multi-Robot Task Allocation in COMSTAR," *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 2007.
[39] Sujit, P. B. and Beard, R., "Distributed Sequential Auctions for Multiple UAV Task Allocation," *Proceedings of the American Control Conference*, 2007.
[40] Zheng, X., Koenig, S., and Tovey, C., "Improving Sequential Single-Item Auctions," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.
[41] Parkes, D. C. and Ungar, L. H., "Iterative Combinatorial Auctions:Theory and Practice," *Proceedings of the 17th National Conference on Artificial Intelligence*, 2000.
[42] Berhault, M., Huang, H., Keskinocak, P., Koenig, S., Elmaghraby, W., Griffin, P., and Kleywegt, A., "Robot Exploration with Combinatorial Auctions," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.
[43] Andersson, A., Tenhunen, M., and Ygge, F., "Integer programming for combinatorial auction winner determination," *Proceedings. Fourth International Conference on MultiAgent Systems*, 2000.
[44] de Vries, S. and Vohra, R., "Combinatorial auctions: A survey," *INFORMS Journal of Computing*, Vol. 15(3), 2003, pp. 284–309.
[45] Rothkopf, M. H., Pekec, A., and Harstad, R. M., "Computationally Manageable Combinatorial Auctions," Tech. rep., Rutgers University, 1998.
[46] Sandholm, T., "Algorithm for Optimal Winner Determination in Combinatorial Auctions," *Artificial Intelligence*, Vol. 135(1-2), 2002, pp. 1–54.
[47] Nandy, M. and Mahanti, A., "An Improved Search Technique for Optimal Winner Determination in Combinatorial Auctions," *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.
[48] Mito, M. and Fujita, S., "On Heuristics for Solving Winner Determination Problem in Combinatorial Auctions," *Journal of Heuristics*, Vol. 2004, 10(5), pp. 507 – 523.
[49] Smith, S. L. and Bullo, F., "Target Assignment for Robotic Networks: Asymptotic Performance under Limited Communication," *Proceedings of the American Control Conference*, 2007.
[50] Smith, S. L. and Bullo, F., "Monotonic Target Assignment for Robotic Networks," *IEEE Trans. on Automatic Control*, Vol. submitted, 2007.
[51] Fujishige, S., "Submodular Functions and Optimization," *Annals of Discrete Mathematics*, Elsevier Science, 1991.
[52] Moore, B. and Passino, K., "Distributed Task Assignment for Mobile Agents," *IEEE Transactions on Automatic Control*, Vol. 52(4), 2007, pp. 749–753.
[53] David, H. and Nagaraja, H., *Order Statistics*, Wiley, 2003.
[54] Wilson, D. B., "Generating Random Spanning Trees More Quickly than the Cover Time," *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing*, 1996.
[55] Brunet, L., Choi, H.-L., and How, J. P., "Consensus-Based Decentralized Auctions for Robust Task Allocation," *IEEE Trans. on Robotics*, Vol. submitted, 2008.

A. Proof of Lemma 1

1. The proof is in two steps: the first is to show that $j \in \mathbf{b}_i^{(n)} \Rightarrow j \in \mathbf{b}_i(t)$ where $\mathbf{b}_i(t)$ is agent i's bundle at the t-th iteration, and the second step is to show $b_{i,k}(t) = b_{i,k}^{(n)},\ \forall k \leq L_i^{(n)}$ where $L_i^{(n)}$ denotes the cardinality of $\mathbf{b}_i^{(n)}$.

   Define the (unordered) set of tasks that have been selected up to n-th SGA step as $\widetilde{\mathcal{J}}_n$ and the set of tasks in agent i's SGA bundle as $\mathcal{J}_{SGA}^i \triangleq \{j : j \in \mathbf{b}_i^{(n)}\}$. By definition of $\mathbf{b}_i^{(n)}$, $\mathcal{J}_{SGA}^i = \{j \in \mathcal{J}_n : z_{i,j} = i\}$ where $\mathcal{J}_n$ is defined by line 7 of SGA. Consider the set of tasks in agent i's CBBA bundle

at iteration t; it is defined as $\mathcal{J}^i_{CBBA} \triangleq \{j \in \mathcal{J} : z_{i,j} = i\}$. Thus, $\mathcal{J}^i_{SGA} \subset \mathcal{J}^i_{CBBA}$ as $\mathcal{J}_n \subset \mathcal{J}$, which equivalently means $j \in \mathbf{b}_i^{(n)} \Rightarrow j \in \mathbf{b}_i(t)$.

The second part of the proof is by induction. Suppose that $\mathbf{b}_i^{(n)} = \{j^\star_{k_1}, \ldots, j^\star_{k_m}\}$ with some $k_1 < \ldots < k_m$. Then, the first entry $j^\star_{k_1}$ is determined from

$$c^{(k_1)}_{i,j^\star_{k_1}} = c_{i,j^\star_{k_1}}(\{\emptyset\}) = \max_{j \in \mathcal{J}_{k_1}} c_{i,j}(\{\emptyset\}), \tag{58}$$

where $\mathcal{J}_{k_1}$ is the reduced task set defined from recursion of line 7 in Algorithm 4, 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

$$J_i = \operatorname*{argmax}_{j \in \mathcal{J}} c_{i,j}(\{\emptyset\}) \times \mathbb{I}(c_{ij}(\{\emptyset\}) \geq y_{ij}) \tag{59}$$

where $\mathbb{I}(\cdot)$ is the indicator function that is unity if the argument is satisfied and zero otherwise. Note that $c_{ij}(\{\emptyset\}) < y_{ij}$ for $j \notin \mathcal{J}_{k_1}$, because all those j's are assigned to the other agents. Thus, the maximization in (59) is equivalent to the maximization in (58) that searches over a more restricted set $\mathcal{J}_{k_1}$. Hence, $c^{(k_1)}_{i,j^\star_{k_1}} \equiv c_{i,J_i}$ and $J_i \equiv j^\star_{k_1}$. Or, the first entry of the CBBA bundle is $j^\star_{k_1}$ that is the first entry of the SGA bundle. Now suppose that the SGA bundle and the CBBA bundle coincide up to the l-th entry. Then, the $(l+1)$-th entry of the SGA bundle $j^\star_{k_{l+1}}$ is determined from

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

where $\mathbf{b}_i^{(1:l)}$ represents the list of the first l entries of agent i's SGA bundle. For $j \notin \mathcal{J}_{k_{l+1}}$, there could be two possibilities: If $j \in \mathbf{b}_i^{(1:l)}$, then $c_{ij} = 0$; otherwise, $y_{ij} > c_{ij}$ because then task j must be in another agent's bundle. Thus,

$$\max_{j \in \mathcal{J}_{k_{l+1}}} c_{i,j}(\mathbf{b}_i^{(1:l)}) \equiv \max_{j \in \mathcal{J}} c_{i,j}(\mathbf{b}_i^{(1:l)}) \times \mathbb{I}(c_{ij} \geq y_{ij}),$$

while the latter finds the $(l+1)$-th entry of the CBBA bundle. Namely, 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 $\mathbf{b}_i^{1:|\mathbf{b}_i^{(n)}|} = \mathbf{b}_i^{(n)}$.

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

$$\max_{j \in \mathcal{J}} c_{i^\star_{n+1},j}(\mathbf{b}^{(n)}_{i^\star_{n+1}}) \times \mathbb{I}(c_{i^\star_{n+1},j}(\mathbf{b}^{(n)}_{i^\star_{n+1}}) \geq y_{i^\star_{n+1},j}).$$

Since $c_{i^\star_{n+1},j}(\mathbf{b}^{(n)}_{i^\star_{n+1}}) = 0$ for $j \in \mathbf{b}^{(n)}_{i^\star_{n+1}}$, and $y_{i^\star_{n+1},j} > c_{i^\star_{n+1},j}(\mathbf{b}^{(n)}_{i^\star_{n+1}})$ for $j \in \mathcal{J}_{n+1} \setminus \mathbf{b}^{(n)}_{i^\star_{n+1}}$, the above maximization is equivalent to

$$\max_{j \in \mathcal{J}_{n+1}} c_{i^\star_{n+1},j}(\mathbf{b}^{(n)}_{i^\star_{n+1}}) \equiv \max_{j \in \mathcal{J}_{n+1}} c^{(n+1)}_{i^\star_{n+1},j},$$

while the right-hand-side expression finds the $(n+1)$-th agent-task pair in SGA. Thus, the solution to the last maximization is $j^\star_{n+1}$ and the corresponding objective value is $c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}}$. Therefore, agent $i^\star_{n+1}$ places a bid of $c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}}$ on task $j^\star_{n+1}$ and locates it at the $(L_i^{(n)}+1)$-th position of its CBBA bundle.

Note that

$$\begin{aligned} c^{(n+1)}_{i^\star_{n+1},j^\star_{n+1}} &\equiv \max_{(i,j) \in \mathcal{I} \times \mathcal{J}} c^{(n+1)}_{ij} = \max_{(i,j) \in \mathcal{I}_{n+1} \times \mathcal{J}} c_{ij}(\mathbf{b}_i^{(n)}) \\ &\geq \max_{i \in \mathcal{I}_{n+1}} c_{ij}(\mathbf{b}_i^{(n)}) \geq \max_{i \in \mathcal{I}_{n+1}} y_{ij}(t) \end{aligned}$$

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

3. The proof is by induction. First, consider the score value for the first SGA assignment:

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

for all $i \in \mathcal{I}$ for all $r \geq 1$ for any $\mathbf{b}$. Since for every $y_{ij}(r)$, $r \geq 1$, there exists k and $\mathbf{b}$ such that $y_{ij}(r) = c_{kj}(\mathbf{b})$, the above relation means $y_{i^\star_1,j^\star_1}(t) = c^{(1)}_{i^\star_1,j^\star_1} \geq y_{i,j^\star_1}(r)$ for all i for all r. $y_{i,j^\star_1}(t)$ can only be changed when some agent places a bid larger than it; the above discussion prevents occurrence of such situations. Therefore, $y_{i,j^\star_1}(s) = y_{i,j^\star_1}(t) = c^{(1)}_{i^\star_1,j^\star_1}$ for all i for all $s \geq t$; this also means $z_{i,j^\star_1}(s) = i^\star_1, \forall i \in \mathcal{I},\ \forall s \geq t$.

Now suppose that $y_{i,j^\star_k}(s) = y_{i,j^\star_k}(t) = c^{(k)}_{i^\star_k,j^\star_k}$, and $z_{i,j^\star_k}(s) = i^\star_k$ for $k \leq m < n$ for all $s \geq t$. Then, as the statement 1 in this lemma holds at iteration s, $\mathbf{b}_i^{1:|\mathbf{b}_i^{(1:m)}|}(s) = \mathbf{b}_i^{(1:m)}$. Think of the $(|\mathbf{b}^{(m)}_{i^\star_{m+1}}| + 1)$-th entry of agent $i^\star_{m+1}$'s bundle. From statement 2 in this lemma, that entry is $\{j^\star_{m+1}\}$ and the corresponding bid is $c^{(m)}_{i^\star_{m+1},j^\star_{m+1}}$, which is the largest score given up to the m-th SGA assignment. This value is identical to the agreed winning bid on $j^\star_{m+1}$ at iteration t, and $y_{i,j^\star_k}$ and $z_{i,j^\star_k}$ for $k \leq m$ are assumed not to change over time; thus, no one can place a bid higher than $c^{(m)}_{i^\star_{m+1},j^\star_{m+1}}$ on task $j^\star_{m+1}$ between t and s. Since the CBBA conflict resolution does not replace a winning bid unless a higher bid shows up, $y_{i,j^\star_{m+1}}$ remains the same after iteration t on, so does $z_{i,j^\star_{m+1}}$. This completes the proof using induction.

4. From statement 3 in this lemma, at any iteration $s \geq t$, (20) 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 bid on $j^\star_{n+1}$ after iteration t; thus, $y_{i^\star_{n+1},j^\star_{n+1}}(s) = y_{i^\star_{n+1},j^\star_{n+1}}(t)$.

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 $i \neq i^\star_{n+1}$ to update $y_{i,j^\star_{n+1}}$ 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 apart at most D hops, every agent will have agreed on the winning bid on task $j^\star_{n+1}$ by iteration $t + D$.

B. Proof of Proposition 2

The multi-assignment problem can be treated as a single assignment with additional combinatorial number of agents. Call agent i's task selection after having selected bundle $\mathbf{b}$ as agent $i^{\mathbf{b}}$; then, there will be a total of $N_u^M \triangleq N_u \cdot \sum_{n=1}^{N_t} N_t!/n!$ agents (because the bundle is an ordered list not unordered set) each of which can only select up to one task. The score for these *expanded* agents can be defined as

$$c_{i^{\mathbf{b}},j} = c_{ij}(\mathbf{b}), \tag{60}$$

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

$$\begin{aligned} &c_{i^{\mathbf{b}},j} = 0, \quad \text{if } j \in \mathbf{b} \\ &c_{i^{\mathbf{b}_1},j} \geq c_{i^{\mathbf{b}_1\oplus\mathbf{b}_2},j}, \quad \forall j \in \mathcal{J},\ \forall \mathbf{b}_1, \mathbf{b}_2. \end{aligned} \tag{61}$$

Similar to the CBAA case, the agent and task indices can be reordered such that

$$i^{\mathbf{b}^{(k)}_{i^\star_k}} = k, \quad j^\star_k = k,\ \forall k \leq N_{\min}. \tag{62}$$

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

$$\mathbf{CBBA} = \sum_{i=1}^{N_{\min}} c_{ii}, \tag{63}$$

with the following conditions being satisfied:

$$\begin{aligned} &c_{ii} \geq c_{kk} \text{ for } k > i \\ &c_{ij} \leq c_{ii},\ c_{ji} \leq c_{ii},\ \text{for } j > i. \end{aligned} \tag{64}$$

Now think of a task swapping procedure for optimality. Like the CBAA case, the high-ranked agent tries to choose a task with smallest loss, while the low-ranked agent tries to pick a task with highest gain. However, for this expanded single assignment case, the task swapping process is more restricted than the CBAA case, because agent i and agent $i^{\mathbf{b}}$ (both in the original indices) cannot independently select their tasks. For instance, suppose that agent i, who has been assigned to task j, picks another task j', agent $i^{\{j\}}$ must release its assignment. Thus, the reselection process is not simply based on pair-wise (or cyclic) task swappings. However, it should be noted that the optimal solution obtained by considering all these restrictions is upper bounded by the *unconstrained swapping* solution that allows inadmissible task swapping as if the expanded single assignment is indeed a single assignment problem. The least favorable situation for the CBBA solution for this unconstrained swapping process is, like CBAA case, where $c_{ij} = c_{ji} = c_{ii}$ for $j > i$. But, since the relations in (61) should be satisfied some of c_{ji}'s corresponding to $j = i^{\mathbf{b}}$ for some $\mathbf{b}$ cannot be equal to c_{ii} but must be zero. Therefore, the performance achieved by unconstrained swapping process on the expanded scores is upper bounded by the same unconstrained swapping process on the score matrix whose zero elements are replaced by maximum possible values. In other words,

$$\begin{aligned} \mathbf{MOPT} &\leq \sum_{i=1}^{\lceil N_{\min}/2 \rceil} c_{ii} + \sum_{i=\lceil N_{\min}/2 \rceil + 1}^{N_{\min}} c_{(N_{\min}-i+1),(N_{\min}-i+1)} \\ &\leq 2 \times \sum_{i=1}^{\lfloor N_{\min}/2 \rfloor} c_{ii} + \sum_{i=\lfloor N_{\min}/2 \rfloor + 1}^{\lceil N_{\min}/2 \rceil} c_{ii} \\ &\leq 2 \times \sum_{i=1}^{N_{\min}} c_{ii} = 2 \cdot \mathbf{CBBA}. \end{aligned} \tag{65}$$

Thus, finally CBBA guarantees 50% optimality.