Journal of Aerospace Information Systems
Vol. 14, No. 2, February 2017

Decentralized Task Allocation Using Local Information Consistency Assumptions

Luke B. Johnson*
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
Han-Lim Choi†
Korea Advanced Institute of Science and Technology, Daejeon 34141, Republic of Korea
Sameera S. Ponda‡
Alphabet, Inc., Mountain View, California 94043
and
Jonathan P. How§
Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

DOI: 10.2514/1.I010461

The information available to agents using decentralized task allocation algorithms plays an important role in how assignments can be constructed. The requirement that information be globally consistent across all agents can be leveraged to allow cooperation on coupled objectives. In environments where global information consistency assumptions are difficult to enforce, the alternative is to rely only on a local best estimate of the global information state, which is referred to here as local information consistency. Algorithms that assume only this local information consistency will have reduced optimization capabilities compared to their global information assumption counterparts. Specifically, if objective functions are non-submodular, local information algorithms may produce arbitrarily bad allocations or, in the case of many algorithms, may not even converge. The key contribution of this paper is an algorithm termed *bid warped consensus-based bundle algorithm* that converges for all deterministic objective functions and has nontrivial performance guarantees for submodular and some non-submodular objective functions. Included in this paper is an analytical analysis of both convergence and performance of the algorithm, as well as a numerical comparison to several other competing local and global information approaches.

I. Introduction

THE goal of standard multiagent task allocation algorithms [1–3] is to coordinate a team of cooperative agents in order to achieve an overall mission objective. These mission objectives can often be broken up into tasks that require specific actions by capable agents, all while satisfying constraints (e.g., fuel, power, vehicle capabilities, etc.). Centralized task allocation algorithms are typically preferred when an application requires high degrees of collaboration. However, in contested environments where communications may be unavailable, unreliable, or have high latency or high cost, relying on centralized solutions may be impractical. In these communication-limited environments, it is necessary to consider distributed or decentralized algorithms [4]. Unfortunately, these architectures introduce additional complications, including difficulties with establishing both algorithmic convergence and performance. The major reason for these complications is that, in decentralized environments, agents may operate on only partial information; thus, independent agent optimizations may not align perfectly with each other. Thus, advanced communication protocols are typically required for decentralized algorithms to optimize over desired objectives. These communication protocols can be considered to use either of two main information assumptions: global information consistency and local information consistency.

1) Global information consistency assumptions (GICAs) require that **all** agents agree upon certain relevant pieces of information during the task allocation algorithm's execution. This agreement forms a set of "correct" information that agents can independently recognize as teamwide truth. Given that these global information consistency assumptions require a teamwide consistency of information, the communication process occurs on a global timescale. Algorithms that use these assumptions can be found in [3,5–23].

2) Local information consistency assumptions (LICAs) do not require global consistency of any information. These algorithms can still use global information but are characterized by only requiring local information consistency (i.e., they do not require any global mechanisms). This approach can provide a much shorter timescale for using new information (as compared to requiring global information consistency) because agents are not required to ensure that this information has propagated to the entire team before using it. The natural downside of this approach is that agents cannot guarantee that any piece of information is globally consistent; thus, algorithms using only local assumptions must be robust to planning with inconsistent information [24–28].

There are many factors that determine whether requiring global or only local information consistency will provide better algorithmic performance for a particular mission. The most important decision variables will be the communication environment, the mission complexity, and the time constraints for creating assignments. A trivial environment where global consistency assumptions would be preferable is an environment where the agents are fully connected with no bandwidth constraints and no mission assignments are time critical. In this environment, there would be no need to introduce an algorithm that handles the added complexity of only relying on local information consistency assumptions. Conversely, a trivial example in which local information consistency assumptions are necessary is when the network can become temporarily disconnected.

Received 3 April 2016; revision received 27 November 2016; accepted for publication 28 November 2016; published online Open Access 24 January 2017.

*Department of Aeronautics and Astronautics, 77 Mass. Ave., Rm. 33-326; lbj16@alum.mit.edu. Member AIAA.

†Associate Professor, Department of Aerospace Engineering, 291 Daehak-ro, Bldg. N7-2, Yuseong; hanlimc@kaist.ac.kr. Member AIAA.

‡Hardware Engineer, X, 1600 Amphitheater Parkway; sponda@alum.mit.edu. Member AIAA.

§Richard Cockburn Maclaurin Professor, Department of Aeronautics and Astronautics, 77 Mass. Ave., Rm. 33-326; jhow@mit.edu. Fellow AIAA.

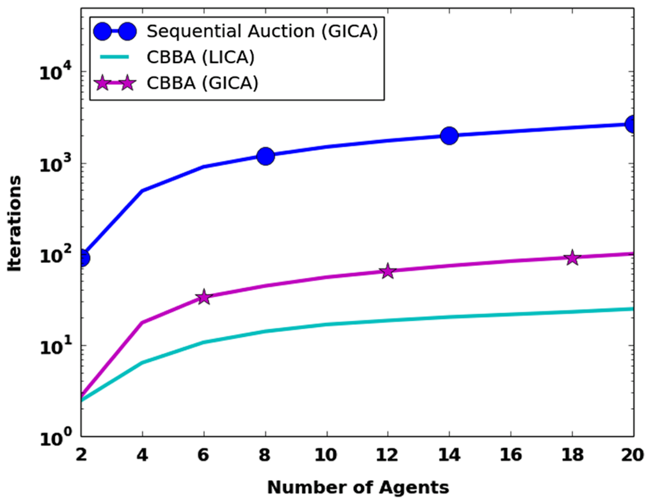


Fig. 1 Comparison of the number of iterations required for convergence in 500 Monte Carlo trials of a 200-task mission with a varying number of agents (see Sec. VI.B for detailed scenarios).

Intermittent network disconnections during global information consistency algorithms will often break the assumptions tied to performance and convergence. Possible repairs would include waiting until the network reconnects (which may never happen) or detecting the new network, continuing on, and hoping that future network dynamics will not break the resulting allocation (which is not a guarantee). Conversely, this domain is handled naturally with local information consistency assumption algorithms.

In other domains where global consistency assumptions are reachable, local consistency algorithms can actually drastically reduce convergence times. Figure 1 shows that the number of iterations to convergence of sequential auction algorithms (such as the one described in [22]) will require roughly the number of tasks times the network diameter number of iterations. Even algorithms that globally consider sets of tasks simultaneously [e.g., a GICA version of the consensus-based bundle *algorithm* (CBBA) [26] that rebroadcasts messages so that each agent has the entire team's bundles during the communication phase] can require significantly more communication iterations than local information consistency assumption algorithms (CBBA; [26]). This is because assignment conflicts are often between agents that are near each other in the communication network, and conflicts can be managed faster using only local conflict-resolution protocols. The full details of this mission scenario, shown in Fig. 1, are provided in Sec. VI.B.

These insights imply that some environments are well suited for algorithms that use only local information consistency assumptions. This paper investigates the limitations that LICA algorithms impose on assignment convergence and performance. In general, issues arise because algorithms that make decisions based only on LICAs cannot trust that their information is globally accurate. This means there is no mechanism to globally enforce assignments, and any local assignment may eventually be replaced. This lack of assignment guarantee can lead to instability in the convergence process. Specifically, if the agent objective functions do not obey a property called submodularity, algorithms that only require LICA may not converge [24–27]. A formal definition for submodular objective functions will be presented in later in this section; but, informally, a submodular objective function requires that the value of servicing a task does not increase due to the assignment of any other tasks. Unfortunately, many objective functions of interest, including those that incorporate fuel penalties, information gathering metrics, cooperative tasking metrics, and stochastic environments may take a non-submodular form.

A solution to this issue is constructed in this paper for some forms of non-submodular functions (see the Appendix for further details about these functional forms). The algorithmic approach, termed bid warped CBBA (BW-CBBA), is to modify the information shared between agents so that it looks as if each agent is using a submodular objective function, even though agent preferences will be determined using non-submodular objectives. This approach provides convergence guarantees for all deterministic objective functions and quantifies when nontrivial performance bounds exist. The main contributions of this paper are 1) presentation of a LICA algorithm that handles more general objectives than available in the literature, and 2) identification of convergence and performance characteristics of this LICA algorithm. In the process, standardized terminology and concepts are introduced for describing LICA algorithms and novel proof techniques are introduced. The preliminary idea for BW-CBBA was first presented in the authors' earlier work [29], but this paper includes a deeper theoretical analysis of the method and a more significant set of numerical comparisons with other approaches.

II. Problem Statement

This section presents the general problem statement and formalizes some of the language and variables used throughout this paper. Given a set of N_a agents and N_t tasks, the goal of the task allocation algorithm is to find a conflict-free matching of tasks to agents that maximizes a global reward. An assignment is said to be conflict free if each task is assigned to no more than one agent. The global objective function for the mission is given by a sum over local objective functions for each agent, whereas each local reward is determined as a function of the tasks assigned to that agent and the times at which those tasks will be serviced. This task assignment problem can be written as the following mixed-integer (possibly nonlinear) program:

$$\max_{x,\tau} \sum_{i=1}^{N_a} \sum_{j=1}^{N_t} F_{ij}(\boldsymbol{x},\boldsymbol{\tau})x_{ij} \tag{1}$$

subject to

$$\boldsymbol{G}(\boldsymbol{x},\tau) \le \boldsymbol{d}$$
$$\boldsymbol{x} \in \{0,1\}^{N_a \times N_t}, \qquad \boldsymbol{\tau} \in \{\mathbb{R}^+\}^{N_a \times N_t}$$

where $\boldsymbol{x} \in \{0,1\}^{N_a \times N_t}$ is a set of $N_a \times N_t$ binary decision variables x_{ij}, which are used to indicate whether or not task j is assigned to agent i; $\boldsymbol{\tau} \in \{\mathbb{R}^+\}^{N_a \times N_t}$ is the set of real-positive decision variables τ_{ij} indicating when agent i will service its assigned task j (where the value of τ_{ij} is irrelevant if task j is not assigned to agent i); F_{ij} is the score function for agent i servicing task j given the overall assignment; and $\boldsymbol{G} = \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{N_c}^T$, with $\boldsymbol{d} = d_1, \ldots, d_{N_c}^T$, defines a set of N_c possibly nonlinear constraints of the form $\boldsymbol{g}_k(\boldsymbol{x}, \boldsymbol{\tau}) \leq d_k$ that captures transition dynamics, resource limitations, tasking assignment constraints, cooperative constraints, etc. This problem formulation can accommodate several different design objectives and constraints commonly used in multiagent decision making problems (e.g., search and surveillance missions where F_{ij} represents the value of acquired information and the constraints $\boldsymbol{g}_k$ capture fuel limitations and/or no-fly zones, or rescue operations where F_{ij} is time critical and favoring earlier τ_{ij} execution times, etc.). An important observation is that, in Eq. (1), the scoring and constraint functions explicitly depend on decision variables $\boldsymbol{x}$ and $\boldsymbol{\tau}$, which makes this programming problem very difficult to solve [nondeterministic polynomial-time (NP) hard] [30].

The shape of the mission objective function is fundamentally related to the difficulty of creating good task assignments. Specifically, objective functions must be submodular [31] for LICA algorithms to converge and have nontrivial performance bounds [24–27]. Define a set function $U(S)$ to be the value of servicing a set of tasks S. Also, then define the marginal score function $U(S|\mathcal{A})$ to be the contribution of adding a set of tasks S to an already existing task allocation $\mathcal{A}$ as $U(S|\mathcal{A}) = U(S \cup \mathcal{A}) - U(\mathcal{A})$. From this, submodularity can be defined as follows:

$$U(S|\mathcal{A}') \geq U(S|\mathcal{A}) \qquad \forall\ A' \text{ subject to } \mathcal{A}' \subset \mathcal{A} \tag{2}$$

A task environment $\mathcal{A}$ can be thought of as a globally consistent set of individual allocations s, where each element $s \in S$ defines an assignment of a single task to a single agent. Equation (2) requires that the value of a particular assignment S cannot increase because of the presence of other assignments. Although many score functions typically used in task allocation satisfy this submodularity condition (for example, mutual information with conditional independence assumptions [32]), many do not. As will be shown in the next section, submodular score functions are essential for most LICA algorithms to guarantee convergence. Previous work identified modifications to the score functions that employed heuristics to ensure that submodularity was satisfied [26], but these heuristics may lead to poor performance and are not usually intuitive to design. This paper presents an online process to modify a LICA algorithm called the consensus-based bundle algorithm that enables the use of non-submodular score functions while still guaranteeing convergence. The principles of the approach are quite general and could be applied to many LICA algorithms.

Recall that the problem statement defined as Eq. (1) uses a score function in addition to some constraints. There are two main ways that these constraints can be encoded into the function analysis of algorithms presented in this paper. Either the constraints can enforce the domain over which the function can be evaluated or infeasible assignments can be defined to have a value of $-\infty$. The analysis of these algorithms is much easier if we can model the objective functions to only be defined over a feasible domain. The approach taken in this paper is to prove that the proposed algorithm will return a feasible solution; then, an analysis can be performed over the feasible domain of the set functions.

III. Non-Submodular Examples

It is simple to demonstrate that a LICA algorithm may fail to converge with a non-submodular score function, even with as few as two tasks and two agents. Consider the following algorithm: each agent sequentially produces bids on a set of available tasks; then, it shares the bids that maximize its local score with the other agents in the team. If an agent bids the highest value for a certain task, it "wins" that task at that round and is allowed to keep it. This process repeats until no agent has incentive to deviate from their chosen allocation. In the following examples, the nominal score achieved for servicing a task will be defined as T. The actual value achieved for servicing the task may be a function of other things the agent has already committed to doing. An agent's bid will be a pair, composed of a task identification (ID) and a task score. The bid represents the information an agent plans to communicate with its neighbors. The notation for an agent's bid in examples 1 and 2 (Figs. 2 and 3, respectively) is (task ID, task score). For the purposes of this algorithm, a bundle is a sequential order of bids, where the later bids are dependent on all earlier bids (in the notation, older bids are on the left).

A. Example 1: Baseline Submodular Score Function

This example (Fig. 2) provides a baseline algorithmic progression with a submodular score function to illustrate how convergence is achieved. In this first example, deviation from a nominal value T will be represented as δ, and its value will be $0 < \delta < T$. In `iteration 1`, each agent chooses between four feasible bundles. The greedy maximum bundle for each agent includes bids on both tasks 1 and 2, but the bid value for the second task placed by both agents is 2δ less than the bid for the first task. The bid values in this example could have been produced by a submodular score function because the score has not increased because of the assignment of the first task (in fact, it has decreased). Between `iteration 1` and `iteration 2`, both agents share their bids with each other and a consistent assignment is reached in `iteration 2` that actually maximizes the global score. When this algorithm is run with submodular score functions, it will return identical allocations to a similar global information consistency version of the algorithm where individual bids are sequentially locked in as a team.

B. Example 2: Non-Submodular Score Function

Example 2 (Fig. 3) highlights how convergence is lost when non-submodular score functions are introduced with the algorithm defined at the beginning of Sec. III. In this example, δ may take any value: $0 < \delta < \infty$. At `iteration 1`, both agents choose between four feasible bundles, and they choose the bundle with the highest total score (agent 1 chooses its bundle 4 and agent 2 chooses its bundle 4). In this example, the maximum value bundle for each agent has a second task that has increased its value because of the assignment of the first task. This explicitly violates the submodularity condition in Eq. (2).

Between `iteration 1` and `iteration 2`, the agents share their bids with each other. Agent 1 is outbid on task 1; thus, the bid on task 2 is invalidated because it depends on task 1 being assigned. Similarly, agent 2 is also outbid on task 2; thus, its bid on task 1 is invalidated. As a result of the conflicts in `iteration 1`, neither agent predicts that it can win either task at `iteration 2`. This includes the fact that neither agent can place a single bid that would outbid their expectation of what the other agent can bid. Thus, for this iteration, neither agent places a bid. Between `iteration 2` and `iteration 3`, each agent will then share their empty bundles. This reverts back to the initial conditions of the algorithm, `iteration 3` repeats `iteration 1`, and the cycle will continue forever.

From example 2, it becomes clear that a LICA algorithm does not work well with non-submodular score functions. It may seem easy as a global observer to see that agent 1 choosing bundle 2 and agent 2 choosing bundle 3 would produce the global optimal objective. However, this requires having the global information of every possible bundle for each agent.

A candidate LICA solution for fixing the convergence issues is to detect cycles of the type shown in example 2 locally, and then use this knowledge to stop suggesting cycling plans. Unfortunately, in general, there is no fast way to detect if a team has entered into one of these cycles

Bundle Creation

`Agent 1 candidate bundles`

1. $\{\}$
2. $\{(1,T)\}$
3. $\{(2,T-\delta)\}$
4. $\{(1,T),(2,T-2\delta)\}$

`Agent 2 candidate bundles`

1. $\{\}$
2. $\{(1,T-\delta)\}$
3. $\{(2,T)\}$
4. $\{(2,T),(1,T-2\delta)\}$

Algorithmic Progression

`Iteration1`

Agent1: $\{(1,T),(2,T-2\delta)\}$

Agent2: $\{(2,T),(1,T-2\delta)\}$

`Iteration2`

Agent1: $\{(1,T)\}$

Agent2: $\{(2,T)\}$

Fig. 2 Example 1: allocations with a submodular score function.

because they could, in general, include a combinatorial number of task assignments being traded between the agents during the consensus process. Even worse is the fact that breaking the algorithm out of one cycle does not guarantee that the agents will not enter another cycle at a later stage in the convergence process.

In practice, many non-submodular score functions can lead to this cycling behavior, and the following example highlights that objective functions that lead to these cycling conditions are not exotic and, in fact, occur for many desirable score functions.

C. Example 3: Waypoint Tasks

Consider the potential task scenario illustrated in Fig. 4 involving one agent (circle labeled $\boldsymbol{a}$) and two tasks (circles labeled 1 and 2). For the purpose of this example, assume that $d_{a2} > d_{a1} \gg d_{12}$, where the notation d_{uv} is the distance required to move from location u to location v. An intuitive score function $F_{ij}(\boldsymbol{x},\boldsymbol{\tau})$ for this environment is defined as follows:

$$F_{ij}(\boldsymbol{x},\boldsymbol{\tau}) = R - f_i d_{x\oplus j} \tag{3}$$

where $F_{ij}(\boldsymbol{x},\boldsymbol{\tau})$ is the score for assigning task j to agent i given current assignment $\boldsymbol{x}$ and arrival timings τ, R is the reward obtained for servicing a task, f_i is the fuel penalty per unit distance for agent i, and $d_{x\oplus j}$ is the increase in distance travelled by inserting task j into the current assignment $\boldsymbol{x}$ (which by assumption cannot already include an assignment on task j by agent i). If the LICA algorithm introduced at the beginning of Sec. III were run in this environment, it would first assign task 1 because $R - f_i d_{a1} > R - f_i d_{a2}$. When the algorithm assigns task 2 using the score function presented in Eq. (3), the score obtained is $R - f_i d_{12}$. This results in the bid on the second task being greater than the first task ($R - f_i d_{12} > R - f_i d_{a1}$), which is exactly the situation shown in example 2 for a non-submodular score function. Depending on bids made by other agents in the fleet, a LICA algorithm may fail to converge with this simple geometry and score function.

One possible strategy to address this problem is to "submodularize" the score function using a heuristic [33]. For example, score function (3) can be approximated as follows:

$$F'_{ij}(\boldsymbol{x},\boldsymbol{\tau}) = R - f_i d_{aj} \tag{4}$$

where the only difference is that the distance metric is defined as the distance d_{aj} measured from the agent's initial location to task j. This is required to ensure that the incremental fuel penalty is never smaller than when the agent has no previous unassigned tasks. With this score function, the first bid will again be on task 1 because $R - f_i d_{a1} > R - f_i d_{a2}$, and the second bid will be on task 2; but, this time, the bid will have the score $R - f_i d_{a2}$. This objective function is now submodular because the score on task 2 does not increase as a result of the previous assignment of task 1.

Bundle Creation

```
Agent 1 candidate bundles
  1. {}
  2. {(1, T)}
  3. {(2, T − δ)}
  4. {(1, T), (2, T + δ)}
Agent 2 candidate bundles
  1. {}
  2. {(1, T − δ)}
  3. {(2, T)}
  4. {(2, T), (1, T + δ)}
```

Algorithmic Progression

```
Iteration 1
    Agent 1: {(1, T), (2, T + δ)}
    Agent 2: {(2, T), (1, T + δ)}
Iteration 2
    Agent 1: {}
    Agent 2: {}
Iteration 3
    Agent 1: {(1, T), (2, T + δ)}
    Agent 2: {(2, T), (1, T + δ)}
```

Fig. 3 Example 2: allocations with a non-submodular score function.

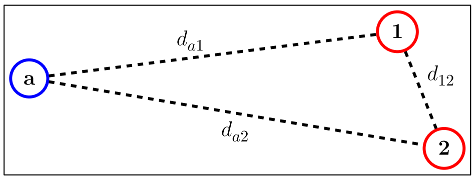


Fig. 4 A potential task environment described in example 3.

However, this score function cannot capture the fact that, because task 1 is being serviced, task 2 should seem much more favorable (as it would have been much closer to the agent after servicing task 1). The purpose of the approach presented in this paper is to enable algorithms with only local information guarantees to use score functions that capture these non-submodular effects without having to sacrifice convergence guarantees. Example missions using both F and F' are explored numerically in Sec. VI, and these demonstrate the potential downside of using a priori submodularized functions.

D. Example 4: Stochastic Tasks

Stochastic objective functions can be used in environments where some necessary planning parameters are not known or even knowable a priori. Examples of these uncertain parameters include wind speed, tolerances on agent performance, duration of search tasks, or other difficult to model a priori environmental effects. When planners use stochastic score functions, combining the score distributions of individual tasks over multiple assignments in a sequence may not have a closed-form solution. In this case, sampling may be required to approximate these distributions. Figure 5 illustrates an example where non-submodular effects can arise due to this necessary sampling. In the figure, the circle with an ***a*** represents the agent, and the circles with **1** and **2** are task locations. The notation t_{uv} is the travel time measured from location u to location v.

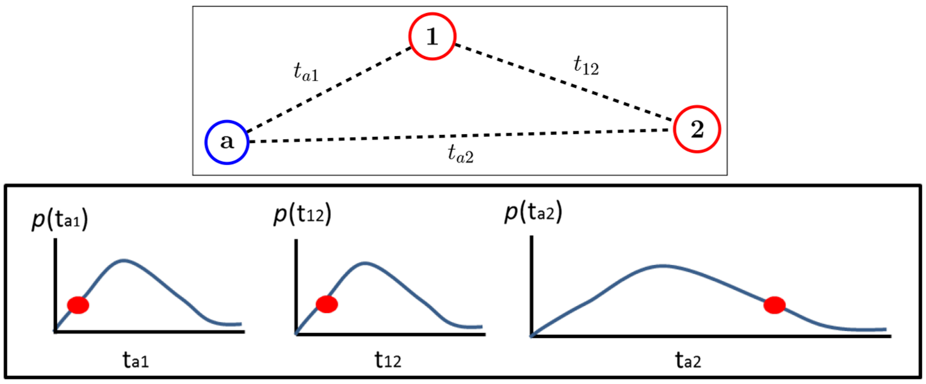


Fig. 5 Potential task environment described in example 4.

Assume that the score function has the following simple form:

$$F_{ij}(\boldsymbol{x}, \boldsymbol{\tau}) = R - \tau_{ij} \tag{5}$$

where τ_{ij} represents the time at which the task is serviced, and R is some fixed reward. This score function then linearly decays with time. This is a submodular score function if the traveling times obey the triangle inequality.

With appropriate distributions over the travel times, it is possible to see that, for a particular sample (Fig. 5), $\boldsymbol{t}_{a2}$ can be greater than the sum of $\boldsymbol{t}_{a1}$ and $\boldsymbol{t}_{12}$:

$$\text{Prob}(\boldsymbol{t}_{a2} > \boldsymbol{t}_{a1} + \boldsymbol{t}_{12}) > 0 \tag{6}$$

Therefore, even though the expected mission scores are submodular,

$$\mathbb{E}(R - \boldsymbol{t}_{a2}) \geq \mathbb{E}(R - (\boldsymbol{t}_{a1} + \boldsymbol{t}_{12})) \tag{7}$$

the sample mean for any finite set of particles of size n sampled from the distributions of $\boldsymbol{t}_{a2}, \boldsymbol{t}_{a1}$, and $\boldsymbol{t}_{12}$ may not be submodular:

$$\text{Prob}\left(\frac{\sum_{m=1}^{n}\left(R - \boldsymbol{t}_{a2}^{[m]}\right)}{n} < \frac{\sum_{m=1}^{n}\left(R - \left(\boldsymbol{t}_{a1}^{[m]} + \boldsymbol{t}_{12}^{[m]}\right)\right)}{n}\right) > 0 \tag{8}$$

In this notation, $\boldsymbol{t}^{[m]}$ is the mth sample from the random variable $\boldsymbol{t}$. Therefore, it is possible for the objective function sample mean for servicing task 2 to be greater by first servicing task 1, which breaks submodularity. This illustrates another example where non-submodularity can arise and must be accounted for to use LICA algorithms in practice.

E. Convergence for Non-Submodular Score Functions

The convergence failures highlighted previously in example 2 are a direct result of multiple tasks being assigned with only local information available about the winners. It was postulated as lemma 4 in [26] that a trick for augmenting the score function to satisfy submodularity would be to ensure that the bids were monotonic in subsequent iterations:

$$\tilde{c}_{ij}(t) = \min\{c_{ij}, \tilde{c}_{ij}(t-1)\} \tag{9}$$

where c_{ij} is the initial score at iteration t, and $\tilde{c}_{ij}(t)$ is the augmented score at iteration t. Unfortunately, this approach tends to create a significant performance degradation in some environments. If this approach is applied to the environment presented in example 2, after `iteration 2`, the algorithm will not be able to bid above zero on either task 1 or 2; thus, no tasks will be assigned for a positive score. This is not a desired result; the approach provided in this paper prevents algorithmic cycling by preemptively changing the bid values rather than relying on the performance-degrading process of identifying cycles after they happen.

F. Using Local Information Consistency Assumption Algorithms

The content of this section outlined how using LICA algorithms can lead to convergence and performance degradation when score functions are non-submodular. Despite this, as was outlined in Sec. I, there exist environments that require or could use LICA task allocation algorithms to improve mission performance. The rest of this paper defines and analyzes algorithmic extensions to an existing LICA algorithm (CBBA) that can increase the class of score functions usable in practice. These modifications allow for the use of LICA task allocation algorithms in many desirable mission domains where performance and convergence guarantees were previously unavailable.

IV. Solution Approach

This section focuses on a new description of the consensus-based bundle algorithm [26] as well as algorithmic modifications specifically related to handling non-submodular score functions. The following description of the CBBA is a slight modification of the one that was originally proposed in [26], introducing some new terms that will be used for proving relevant aspects of the new algorithm proposed in this paper.

Algorithm 1 CBBA: bundle building phase

```
(for agent $i$)
1: Procedure Build Bundle ($\mathcal{A}_i$)
2:   for all $s_{ij'}$s.t.$\exists j'$ where $s_{ij'} \in \mathcal{A}_i$, do
3:     $\mathcal{A}_i \leftarrow \mathcal{A}_i \setminus s_{ij'}$
4:   end for
5:   set $\boldsymbol{b}_i \leftarrow \varnothing$
6:   while $|b_i < L_t|$, do
7:     $\mathcal{J} \leftarrow \{j | s_{ij} \notin \mathcal{A}_i\}$
8:     for all $j \in \mathcal{J}$ do
9:       $c_{ij} \leftarrow F_{ij}(\boldsymbol{b}_i)$,
10:      $h_{ij} \leftarrow \prod_{s_{i'j} \in \mathcal{A}_i} \mathbb{I}(c_{ij} > c_{i'j})$
11:    end for
12:    $j^\star \leftarrow \arg\max_{j \in \mathcal{J}} c_{ij} \cdot h_{ij}$
13:    $s_{ij^\star} \leftarrow \langle i, j^\star, c_{ij^\star} \rangle$
14:    if $c_{ij^\star} \cdot h_{ij^\star} > 0$, then
15:      $\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i \oplus s_{ij^\star}$
16:      $\mathcal{A}_i \leftarrow \mathcal{A}_i \oplus s_{ij^\star}$
17:    else
18:      break
19:    end if
20:  end while
21:  return $(\boldsymbol{b}_i, \mathcal{A}_i)$
22: end procedure
```

A. Baseline CBBA

The CBBA is a local information consistency assumption auction algorithm. The algorithmic structure of the CBBA is an iterative two-phase algorithm. These two phases are a bundle building phase where each vehicle greedily generates an ordered list of assignments and a task consensus phase where conflicting assignments are identified and resolved through local communication between neighboring agents. These two phases are repeated until the algorithm has reached convergence. To further explain the relevant details of the algorithm, some notation will first be formalized.

1) A bid is represented as a triple, $s_{ij} = \langle i, j, c_{ij} \rangle$, where i represents the bidding agent's index, j represents the task's index, and c_{ij} represents the value of assignment for this task agent pair.

2) A bundle is an ordered data structure internal to each agent i, $\boldsymbol{b}_i = (s_{ij_1}, \ldots, s_{ij_n})$, that consists of a list of bids where s_{ij_k} is the kth bid added to the bundle. A bundle is said to have length n if there are n bids in the list. When new bids are added to the bundle, they are appended to the end; thus, the order in the bundle reflects the relative age of each bid, and thus the dependency structure of the bids.

3) The bid space is an unordered set of bids, defined as $\mathcal{A} = \{s_{i_1 j_1}, \ldots, s_{i_N j_N}\}$, where N is defined to be the current size of the bid space. This bid space contains a globally consistent set of the current winning bids in the team.

4) A local bid space $\mathcal{A}_i$ is defined as a set that contains agent i's current local understanding of the global bid space. In a fully connected network, $\mathcal{A}_i = \mathcal{A}$ after each task consensus phase (which also would correspond to having global information consistency assumptions over the task space); but, in general, the geometry of agents in the network may lead to information propagation latencies, and thus nonidentical local bid spaces. A consistent global bid space will be always be a subset of the local bid spaces $\mathcal{A} \subseteq \mathcal{A}_i$.

5) The network diameter $\mathcal{D}$ is defined as the number of communication hops between the furthest agent pair in the communication network. More formally, define a number for each agent i consisting of the minimum communication distance to every other agent i'. The maximum value over all agents is defined as the network diameter.

The CBBA begins with each agent i being provided (or somehow discovering) a set of available tasks. In general, the set of available tasks does not need to be identical for all agents. The two-phase algorithm then begins in the bundle building.

1. Bundle Building Phase

For each agent i, the bundle building phase is run independently.

1) All current tasks that agent i has won are removed from agent i's bundle $\boldsymbol{b}_i$ and local bid space $\mathcal{A}_i$ (lines 3 and 5 of Algorithm 1). This step is required for the performance guarantees of the algorithm[¶] but, in most cases, the agent will re-add each of the tasks it has just dropped.

2) A local internal score function $F_{ij}(\boldsymbol{b}_i)$ is defined for each agent i and task j. It is a function of the agent's current bundle $\boldsymbol{b}_i$ and implicitly a function of the assignment constraints $\boldsymbol{G}(\boldsymbol{x}, \tau) \le \boldsymbol{d}$ posed in the problem formulation [Eq. (1)]. If a proposed assignment will not satisfy the constraints required by Eq. (1), $F_{ij}(\boldsymbol{b}_i)$ will return a value of $-\infty$. For complete notation consistency with Eq. (1) in Sec. II, assume that there is a one-to-one mapping between $(\boldsymbol{b}_i)$ and $(\boldsymbol{x}, \tau)$. The complexity of actually picking the execution times τ_{ij} is not a focus of this paper, but other solutions used for the CBBA can be implemented [34]. For each task j available in the environment (line 7), each agent i uses its local internal score function $F_{ij}(\boldsymbol{b}_i)$ to create a score c_{ij} (line 9).

3) These scores c_{ij} are compared with the winning bid information for the corresponding task j located in the agent's local bid space $\mathcal{A}_i$ (line 10). The largest score that would outbid the current winner in the local bid space is chosen as agent i's next bid (line 12). A bid $s_{ij^\star}$ is created (line 13) and, as long as the value of the bid is positive $c_{ij^\star} \cdot h_{ij^\star} > 0$ (line 14), it is placed at the end of the bundle (line 15) and is added as the winning bid on task j in the local bid space $\mathcal{A}_i$ (line 16).

4) Steps 2 and 3 (lines 7–16) are repeated until no tasks have a larger score than the corresponding bids already in $\mathcal{A}_i$ or the maximum bundle length L_t (line 6) is reached: at which point, the bundle building phase terminates. In this formulation, the values c_{ij} that are used to rank the tasks are the same as the values used to construct the bids s_{ij}. The algorithm constructed later in this paper separates these two values in order to provide convergence for all objective functions as well as nontrivial performance guarantees for some classes of non-submodular objective functions.

[¶]Agent i may want to change its bids in light of new information obtained through communication instead of being "stuck" with the bids made in the previous iteration.

2. Task Consensus Phase

After the bundle building phase completes, each agent i synchronously shares its current local bid space $\mathcal{A}_i$ with each of its adjacent neighbors. This local bid space, in combination with timestamp information, is then passed through a decision table (see [26], table 1, for details) that provides all of the conflict resolution logic to merge local bid spaces. In general, the consensus logic prefers larger and more recent bids. If the consensus phase has occurred more than twice the network diameter times without any bids changing, the algorithm has converged and terminates; if not, each agent reenters the bundle building phase and the algorithm continues.

3. Score Function

Fundamental to all of the convergence and performance guarantees for the CBBA is that it must use diminishing marginal gains (DMGs) to satisfy the score function. The requirement of DMGs for the CBBA score function is a special case of requiring submodularity, as was introduced in Sec. I, because it was defined for a specific marginal contribution to the existing bundle $\boldsymbol{b}_i$ as opposed to for all sets as was defined in Eq. (2). It was recognized in the seminal description of the CBBA [26] but is updated here with the notation of bids and bundles. The DMG is defined as

$$F_{ij}(\boldsymbol{b}_i) \geq F_{ij}(\boldsymbol{b}_i \oplus_{\text{end}} s_{ij'}) \quad \forall\ j' \neq j \tag{10}$$

where $\boldsymbol{b}_i \oplus_{\text{end}} s_{ij'}$ refers to adding a bid $s_{ij'}$ on task j' to an already existing bundle $\boldsymbol{b}_i$. Roughly, this condition means that no bids $s_{ij'}$ can be made on any other task j' that would increase c_{ij}, which is agent i's score for task j. When score functions $F_{ij}(\boldsymbol{b}_i)$ are defined as the marginal contribution of adding a bid on task j to an existing bundle $\boldsymbol{b}_i$ (which is what is done in this paper), the submodularity constraint can replace requiring DMGs.

B. Bid Warping

The approach presented in this section changes two fundamental aspects of placing bids. First, the ranking of task scores is allowed to use an objective function that does not satisfy submodularity and the external bid values (those shared with other agents) are not identical to the internal scores used for deciding which are the highest-value tasks. To highlight the algorithmic changes, some additional notation is needed.

Bid warping uses the internal score c_{ij} and the current bundle $\boldsymbol{b}_i$ to construct a warped score $\bar{c}_{ij}$:

$$\bar{c}_{ij} = \min\left\{c_{ij}, \min_{k\in\{1,\ldots,|b_i|\}} c_{ij_k}\right\} \tag{11}$$

where c_{ij_k} is the unwarped score of the kth element in the current bundle, and $|\boldsymbol{b}_i|$ is the length of the current bundle. The warped score can also be recursively defined as the minimum of the unwarped score c_{ij} and the value of the most recently warped bid added to the bundle $\bar{c}_{ij_{|b_i|}}$

$$\bar{c}_{ij} = \min\left\{c_{ij}, \bar{c}_{ij_{|b_i|}}\right\} \tag{12}$$

Definition 1: Define a strict bid ordering. In this paper, the tie breaker will be defined to be the lowest agent ID. Therefore, for bids $s_{ij} = \langle i, j, c_{ij}\rangle$,

$$s_{i_1j_1} \succ s_{i_2j_2} \Rightarrow c_{i_1j_1} > c_{i_2j_2} \tag{13}$$

$$\vee\ c_{i_1j_1} = c_{i_2j_2} \quad \text{and} \quad i_1 < i_2 \tag{14}$$

$$\vee\ c_{i_1j_1} = c_{i_2j_2} \quad \text{and} \quad i_1 = i_2 \quad \text{and} \quad s_{i_1j_1}\ \text{earlier in bundle than}\ s_{i_2j_2} \tag{15}$$

These three conditions can completely define a strict ordering over bids. The first "or" clause [Eq. (13)] defines that, if the score $c_{i_1j_1}$ of $s_{i_1j_1}$ is larger than the score $c_{i_2j_2}$ of $s_{i_2j_2}$, then $s_{i_1j_1} \succ s_{i_2j_2}$. The second or clause [Eq. (14)] defines that, if the scores are the same $c_{i_1j_1} = c_{i_2j_2}$, then the bid with the lowest agent ID is larger. The third or clause [Eq. (15)] is reached when the scores are the same and the agent IDs are the same. This is the case when the two bids are in a single agent's bundle, so it is defined that the earliest element in the bundle is larger.

C. Bid Warped CBBA

This section presents the main algorithmic modifications required for the CBBA to use non-submodular score functions, the result of which is called bid warped CBBA.

1. Bid Warped CBBA

This algorithm (Algorithm 2: `BW-CBBA`) is run independently on each agent i and is initialized each time the team decides to replan (or construct an initial allocation).

1) A BW-CBBA assignment iteration is initialized with agent i's old bundle $\boldsymbol{b}_i^o$ and its local understanding of the global bid space $\mathcal{A}_i$. To initialize the algorithm, the convergence counter for agent i, denoted by k_i, is set to zero (line 2 in Algorithm 2); its broadcast queue Q_i, which is a list of pairs $\langle s, t\rangle$ consisting of a bid s and a timestamp t, is set to empty (line 3); and its local timestamp matrix $\mathcal{Z}_i(i', j)$, which records the timestamp of the most recent information about a bid made on task j by agent i', is initialized to all zeros (line 4).

2) Although this algorithm has not converged (defined as when the size of the broadcast queue $|Q_i| = 0$ for $2\mathcal{D}$ iterations at line 5), the algorithm iterates between running a bundle building phase (`BW-BB` at line 11) and a task consensus phase (`BW-TC` at line 12).

Algorithm 2 BW-CBBA: bid warped CBBA

```
1: procedure BW-CBBA ($\boldsymbol{b}_i^o, \mathcal{A}_i$)
2:   $k_i \leftarrow 0$
3:   $Q_i \leftarrow \{\}$
4:   $\mathcal{Z}_i \leftarrow zeros(N_a, N_t)$
5:   while $k_i < 2 \cdot \mathcal{D}$, do
6:     if $|Q_i| = 0$, then
7:       $k_i \leftarrow k_i + 1$
8:     else
9:       $k_i \leftarrow 0$
10:    end if
11:    $\boldsymbol{b}_i, \mathcal{A}_i \leftarrow$ BW-BB ($\mathcal{A}_i$)
12:    $\mathcal{A}_i, Q_i, \mathcal{Z}_i \leftarrow$ BW-TC ($\boldsymbol{b}_i^o, \boldsymbol{b}_i, \mathcal{A}_i, Q_i, \mathcal{Z}_i$)
13:    $\boldsymbol{b}_i^o \leftarrow \boldsymbol{b}_i$
14:  end while
15:  return ($\mathcal{A}_i$)
16: end procedure
```

Algorithm 3 BW-BB: bid warped bundle building

```
1: procedure BW-BB ($\mathcal{A}_i$)
2:   for all $\bar{s}_{ij'}$ s.t. $\exists j'$ where $\bar{s}_{ij'} \in \mathcal{A}_i$, do
3:     $\mathcal{A}_i \leftarrow \mathcal{A}_i \setminus \bar{s}_{ij'}$
4:   end for
5:   $\boldsymbol{b}_i \leftarrow \emptyset$
6:   while $|\boldsymbol{b}_i| < L_t$, do
7:     $\mathcal{J} \leftarrow \{j | \bar{s}_{ij} \notin \mathcal{A}_i\}$
8:     for all $j \in \mathcal{J}$, do
9:       $c_{ij} \leftarrow F_{ij}(\boldsymbol{b}_i)$
10:      $\bar{c}_{ij} \leftarrow \min\{c_{ij}, \bar{c}_{ij_{|b_i|}}\}$ [Eq. (12)]
11:      $h_{ij} \leftarrow \prod_{\bar{s}_{i'j} \in \mathcal{A}_i} \mathbb{I}(\bar{c}_{ij} > \bar{c}_{i'j})$
12:    end for
13:    $j^\star \leftarrow \underset{j \in \mathcal{J}}{\text{argmax}}(c_{ij} \cdot h_{ij})$
14:    $\bar{s}_{ij^\star} \leftarrow \langle i, j^\star, \bar{c}_{ij^\star} \rangle$
15:    if $\bar{c}_{ij^\star} \cdot h_{ij^\star} > 0$, then
16:      $\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i \oplus \bar{s}_{ij^\star}$
17:      $\mathcal{A}_i \leftarrow \mathcal{A}_i \oplus \bar{s}_{ij^\star}$
18:    else
19:      break
20:    end if
21:  end while
22:  return ($\boldsymbol{b}_i, \mathcal{A}_i$)
23: end procedure
```

2. *Bid Warped Bundle Building*

Again, for each agent i, the bundle building phase is run independently (Algorithm 3: BW-BB).

1) All current tasks in agent i's bundle $\boldsymbol{b}_i$ and tasks won by agent i in its local bid space $\mathcal{A}_i$ (lines 3 and 5 in Algorithm 3) are removed.

2) Define a set of available tasks $\mathcal{J}$ to be those that are not already in agent i's local bid space $\mathcal{A}_i$ (line 7).

3) For each task $j \in \mathcal{J}$, each agent i uses its local internal score function $c_{ij} \leftarrow F_{ij}(\boldsymbol{b}_i)$, which is a function of its current bundle, to create a score c_{ij} (line 9). Again, $F_{ij}(\boldsymbol{b}_i)$ is implicitly a function of the assignment constraints $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{\tau}) \leq \boldsymbol{d}$ posed in the problem formulation [Eq. (1)] and, if a proposed assignment will not satisfy the constraints required by Eq. (1), $F_{ij}(\boldsymbol{b}_i)$ will return a value of $-\infty$. The only other requirement on the score function F_{ij} in this formulation is that, for each agent i, the returned scores must be repeatable. In this context, being repeatable means that, conditional on an identical bundle and set of constraints, the function returns an identical score.

4) The score values c_{ij} are then warped using Eq. (12) (line 10).

5) Each of the warped bid values $\bar{c}_{ij}$ is compared with the winning bid values for the corresponding task j located in the local bid space $\mathcal{A}_i$ to create an indicator function defining if agent i can outbid the current winner of task j with its warped bid (line 11). The task $j^\star$ with the largest original score c_{ij}, for which the warped bid $\bar{c}_{ij}$ would outbid the current winner in the local bid space, is chosen as agent i's next bid (line 13). A warped bid $\bar{s}_{ij^\star}$ is created (line 14) and, as long as the value of the warped bid is positive $\bar{c}_{ij^\star} \cdot h_{ij^\star} > 0$ (line 15), it is placed at the end of the bundle (line 16), and it replaces the current bid on task j in the local bid space $\mathcal{A}_i$ (line 17).

6) If no bids are able to be outbid in $\mathcal{A}_i$ or the maximum bundle length L_i is reached, the bundle building phase terminates; if not, steps 2–5 are repeated. The key insight in this algorithm is that the value c_{ij} is used to rank the bids but the warped bid $\bar{s}_{ij}$ is what is actually shared with the other agents and is what is used to determine if a bid is able to overbid what is already in the bid space $\mathcal{A}_i$.

3. *Bid Warped Task Consensus*

The purpose of this function (Algorithm 4: BW-TC) is to allow agents to exchange task assignment information with neighboring agents. The procedure is run independently for each agent, but neighboring agents synchronize their `broadcast` (line 3) and `receive messages` (line 4) steps.

Algorithm 4 BW-TC: bid warped task consensus

```
1: procedure BW-TC ($\boldsymbol{b}_i^o, \boldsymbol{b}_i, \mathcal{A}_i, Q_i, \mathcal{Z}_i$)
2:   $Q_i, \mathcal{Z}_i \leftarrow$ BW-TC-UQBC ($Q_i, \mathcal{Z}_i, \boldsymbol{b}_i^o, \boldsymbol{b}_i$)
3:   Broadcast ($Q_i$)
4:   $\mathcal{M}_i \leftarrow$ ReceiveMessages ()
5:   $\mathcal{A}_i, Q_i, \mathcal{Z}_i \leftarrow$ BW-TC-PRM ($\mathcal{M}_i, \mathcal{A}_i, Q_i, \mathcal{Z}_i$)
6:   return ($\mathcal{A}_i, Q_i, \mathcal{Z}_i$)
7: end procedure
```

Algorithm 5 BW-TC-UQBC: bid warped task consensus update queue with bundle changes

```
 1: procedure BW-TC-UQBC ($Q_i$, $\mathcal{Z}_i$, $\boldsymbol{b}_i^o$, $\boldsymbol{b}_i$)
 2:   for all $\{\bar{s}_{ij}^o | \bar{s}_{ij}^o \in \boldsymbol{b}_i^o, \bar{s}_{ij}^o \notin \boldsymbol{b}_i\}$, do
 3:     $Q_i \leftarrow Q_i \cup \langle \text{Dropbid}(\bar{s}_{ij}^o), t_{now} \rangle$
 4:     $\mathcal{Z}_i(i, j) \leftarrow t_{now}$
 5:   end for
 6:   for all $\{\bar{s}_{ij} | \bar{s}_{ij} \in \boldsymbol{b}_i, \bar{s}_{ij} \notin \boldsymbol{b}_i^o\}$, do
 7:     $Q_i \leftarrow Q_i \cup \langle \bar{s}_{ij}, t_{now} + \delta \rangle$
 8:     $\mathcal{Z}_i(i, j) \leftarrow t_{now} + \delta$
 9:   end for
10:   return ($Q_i, \mathcal{Z}_i$)
11: end procedure
```

1) This algorithm first updates the broadcast queue Q_i and the timestamp information for agent i $\mathcal{Z}_i(i, j)$ through the `BW-TC-UQBC` function (line 2) by accounting for the changes between agent i's old bundle $\boldsymbol{b}_i^o$ and its new bundle $\boldsymbol{b}_i$. The details of this procedure are defined as in Algorithm 5.

2) The newly updated queue Q_i is then broadcast to agent i's network neighbors (line 3) using function `broadcast` (Q_i). Grouping sets of messages together (as opposed to sending information out incrementally) is necessary because message groupings define a consistent information state from the sending agent (i.e., some bids only make sense with the existence of earlier drop bids, etc.).

3) The information received by each agent i, via the broadcasts from its neighbors, is collected as $\mathcal{M}_i$ (line 4) using the function $\mathcal{M}_i \leftarrow$ `RecieveMessages()`. It is worth noting that the message set received by each agent may be different if the network is not strongly connected.

4) The function `BW-TC-PRM` (line 5) is then called with the purpose of updating local information ($\mathcal{A}_i, Q_i, \mathcal{Z}_i$) in response to the received messages $\mathcal{M}_i$. The details of this procedure are presented as in Algorithm 6.

4. Bid Warped Task Consensus Update Queue with Bundle Changes

The purpose of this function (Algorithm 5: `BW-TC-UQBC`) is to update the broadcast queue Q_i and local timestamp matrix $\mathcal{Z}_i$ with the local changes made to agent i's bundle $\boldsymbol{b}_i$ in the bundle building phase (Algorithm 3).

1) This algorithm first searches over all bids $\bar{s}_{ij}^o$ that were in the old bundle $\boldsymbol{b}_i^o$ but not in the new bundle $\boldsymbol{b}_i$ (line 2). A drop bid is then created for all of these bids (Dropbid($\bar{s}_{ij}^o$)) and added to the broadcast queue Q_i (line 3). A drop bid is defined as a bid that signifies the removal of a previously placed bid. The function Dropbid(s) returns a drop bid corresponding to bid s. Correspondingly, the timestamp element for this task is also updated in $\mathcal{Z}_i$ (line 4).

2) Similarly, the algorithm then searches over all new bids $\bar{s}_{ij}$ that are in the new bundle $\boldsymbol{b}_i$ but not in the old bundle $\boldsymbol{b}_i^o$ (line 6). Each of these new bids $\bar{s}_{ij}$ (with a timestamp) is added to the broadcast queue Q_i (line 7) and the local timestamp matrix $\mathcal{Z}_i$ (line 8) with the current time t_{now}, plus a small extra value called δ. It is important that the timestamps introduced here are larger by this small margin δ to ensure that other agents can infer that these new bids are later than potentially created dropped bids from earlier in the function at line 3.

5. Bid Warped Task Consensus Process Received Messages

The purpose of this algorithm (Algorithm 6: `BW-TC-PRM`) is to update the local planning knowledge of agent i in response to messages received $\mathcal{M}_i$. The local knowledge updated includes agent i's local bid space $\mathcal{A}_i$, its broadcast queue Q_i, and its local timestamp matrix $\mathcal{Z}_i$.

1) This function first searches through each message $\langle \bar{s}_{i_m j_m}, t_m \rangle$ in $\mathcal{M}_i$ to find those that are drop bids (lines 2–3). If the drop bid is new $\mathcal{Z}_i(i_m, j_m) < t_m$ (line 4), update the timestamp matrix (line 5), add the drop bid to the rebroadcast queue Q_i (line 6), and if there is a bid $\bar{s}_{i_m j_m}$ in agent i's local bid space $\mathcal{A}_i$ (line 7) created by agent i_m on task j_m, then remove it from the bid space $\mathcal{A}_i$ (line 8).

2) This function then iterates through each message $\langle \bar{s}_{i_m j_m}, t_m \rangle$ in $\mathcal{M}_i$ that is not a drop bid (lines 13 and 14). Again, if it is a new bid (line 15), update the timestamp matrix for agent i_m and task j_m. If there is not a bid in agent i's local bid space $\mathcal{A}_i$ on task j_m, then add the bid message to the bid space (line 18) and add the bid and its corresponding timestamp to the broadcast queue Q_i (line 19). Otherwise, there is a bid on task j_m in the local bid space, so assign the bid in the local bid space to the name $\bar{s}'_{i' j_m}$ (line 21). If the bid message $\bar{s}_{i_m j_m}$ is greater than the bid that is currently in the bid space $\bar{s}'_{i' j_m}$ (line 22), then remove the old bid from the bid space (line 23), add the new bid message to the bid space (line 24), and add the new bid message to the broadcast queue (line 25). If the bid message does not outbid the local bid, then add the local bid to the broadcast queue with its corresponding timestamp that is stored as $\mathcal{Z}_i(i', j_m)$ (line 27).

D. Comparison to CBBA Consensus Phase

The task consensus phase presented here is differs from the one that was presented in [26]. The previously published consensus phase requires a rebroadcast of every task at every iteration. Therefore, as long as messages can be assumed to be delivered completely (and the network can be assumed to remain connected), the approach presented in this paper will use less messaging overall. However, this approach is less robust to dropped messages or networks that change topology on the timescale of the plan convergence time. In these domains, some information may never reach parts of the network. If the messaging channels are not reliable and messages are not guaranteed to arrive, then the approach presented in [26]

Algorithm 6 BW-TC-PRM: bid warped task consensus process received messages

```
1: procedure BW-TC-PRM $\mathcal{M}_i, \mathcal{A}_i, \mathcal{Z}_i, Q_i$
2:   for all $\langle \bar{s}_{i_m j_m}, t_m \rangle \in \mathcal{M}_i$, do
3:     if $\bar{s}_{i_m j_m}$ is a drop bid, then
4:       if $\mathcal{Z}_i(i_m, j_m) < t_m$, then
5:         $\mathcal{Z}_i(i_m, j_m) \leftarrow t_m$
6:         $Q_i \leftarrow Q_i \cup \langle \bar{s}_{i_m j_m}, t_m \rangle$
7:         if $\bar{s}_{i_m j_m} \in \mathcal{A}_i$, then
8:           $\mathcal{A}_i \leftarrow \mathcal{A}_i \setminus \bar{s}_{i_m j_m}$
9:         end if
10:      end if
11:    end if
12:  end for
13:  for all $\langle \bar{s}_{i_m j_m}, t_m \rangle \in \mathcal{M}_i$, do
14:    if $\bar{s}_{i_m j_m}$ is not a drop bid, then
15:      if $\mathcal{Z}_i(i_m, j_m) < t_m$, then
16:        $\mathcal{Z}_i(i_m, j_m) \leftarrow t_m$
17:        if $\forall\ i', \bar{s}'_{i' j_m} \notin \mathcal{A}_i$, then
18:          $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \bar{s}_{i_m j_m}$
19:          $Q_i \leftarrow Q_i \cup \langle \bar{s}_{i_m j_m}, t_m \rangle$
20:        else
21:          $\bar{s}'_{i' j_m} \leftarrow \langle i', j_m, \bar{c}' \rangle \in \mathcal{A}_i$
22:          if $\bar{s}_{i_m j_m} \succ \bar{s}'_{i' j_m}$, then
23:            $\mathcal{A}_i \leftarrow \mathcal{A}_i \setminus \bar{s}'_{i' j_m}$
24:            $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \bar{s}_{i_m j_m}$
25:            $Q_i \leftarrow Q_i \cup \langle \bar{s}_{i_m j_m}, t_m \rangle$
26:          else
27:            $Q_i \leftarrow Q_i \cup \langle \bar{s}'_{i' j_m}, \mathcal{Z}_i(i', j_m) \rangle$
28:          end if
29:        end if
30:      end if
31:    end if
32:  end for
33:  return $(\mathcal{A}_i, Q_i, \mathcal{Z}_i)$
34: end procedure
```

should be used. The following performance and convergence guarantees assume the consensus protocol defined in this paper, but the same results with different notation and approach could be obtained by using an alternative consensus protocol like the one defined in table 1 of [26].

V. Convergence and Performance Characteristics of BW-CBBA

This section proves the two main theoretical results of the paper:

1) BW-CBBA converges to a teamwide consistent solution in, at most, $2N_t\mathcal{D}$ iterations.

2) BW-CBBA achieves nontrivial performance bounds for some classes of objective functions. The only objective function assumption needed for convergence of BW-CBBA is that, given identical initial conditions (local bid space $\mathcal{A}_i$ and bundle $\boldsymbol{b}_i$), the subsequent bid values produced are repeatable. Again, this condition allows for score functions that are stochastic, but evaluating relevant metrics that decide bid ordering over the stochastic distributions must be repeatable.

The proofs presented in this section do not follow the styles presented in [26] for two main reasons:

1) BW-CBBA does not return the same solution as a centralized sequential greedy solver in all cases (as the CBBA proof had assumed).

2) Using a proof by construction approach for the convergence analysis provides insight into the algorithmic progression and is better suited as an analytical tool when evaluating potential future modifications to BW-CBBA.

A last note about this section is that the performance proof returns the same bound as presented in [26] when the internal score function F is submodular and monotonic.

A. Convergence Guarantee

Lemma 1: The values of the warped bids $\bar{s}_{ij}$ (line 14 of Algorithm 3) added to bundles $\boldsymbol{b}_i$ have a monotonically decreasing ordering:

$$\bar{s}_{ij_k} \succ \bar{s}_{ij_{k+1}} \quad \forall\ k \in \{1, \ldots, |\boldsymbol{b}_i| - 1\}$$

where $\bar{s}_{ij_k}$ is the kth bid added to agent i's bundle.

Proof: According to the definition of the bid warping [Eq. (12)], the warped bid values are defined as

$$\bar{c}_{ij_{k+1}} = \min\{c_{ij_{k+1}}, \bar{c}_{ij_k}\}$$

forcing $\bar{c}_{ij_k} \geq \bar{c}_{ij_{k+1}}$. Two conditions can then arise: 1) $\bar{c}_{ij_k} > \bar{c}_{ij_{k+1}}$, and therefore $\bar{s}_{ij_k} \succ \bar{s}_{ij_{k+1}}$ from Eq. (13) of Definition 1; and 2) $\bar{c}_{ij_k} = \bar{c}_{ij_{k+1}}$ but, because $(k < k + 1)$, $\bar{s}_{ij_k}$ is located earlier in the bundle than $\bar{s}_{ij_{k+1}}$. From Eq. (15) of Definition 1, $\bar{s}_{ij_k} \succ \bar{s}_{ij_{k+1}}$.

Note that, to reduce the notation clutter, all subsequent overbars will be removed, but all bids discussed in this Convergence Guarantee section (Sec. V.A) will be considered to be warped bids.

Definition 2: Define an overbid $s'_{i'j'}$ to be a bid that agent i receives via communication from other agents that change its local bid space $\mathcal{A}_i$ (Algorithm 6). An overbid can also be a drop bid, which was previously defined as a bid message specifying the removal of a previous bid. Drop bids are defined to have the same relative size of the bid they correspond to dropping (via Algorithm 6, line 8), where the size of the bid is defined in Definition 1.

Definition 3: Define $\sigma_{i_\sigma j_\sigma}(S')$ to be the largest bid in a set of overbids S' that is received by agent i:

$$\sigma_{i_\sigma j_\sigma}(S') = \max_{s'_{i'j'} \in S'} s'_{i'j'}$$

Theorem 1: After an agent i receives a set of overbids S', all bids larger than the largest overbid $\sigma_{i_\sigma j_\sigma}(S')$ will remain unchanged, i.e., all bids $s_{ij} \in \boldsymbol{b}_i$ subject to

$$s_{ij} \succ \sigma_{i_\sigma j_\sigma}(S') \tag{16}$$

will remain in agent i's bundle $\boldsymbol{b}_i$ at the next bundle building iteration.

Proof: The form of this proof will be to first show that bids smaller than $\sigma_{i_\sigma j_\sigma}(S')$ cannot affect the assignment of larger bids, and thus show that bids larger than $\sigma_{i_\sigma j_\sigma}(S')$ will not be dropped. First assume that $\exists s^{\ominus}_{ij^{\ominus}} \in \boldsymbol{b}_i$ subject to $s^{\ominus}_{ij^{\ominus}} \prec \sigma_{i_\sigma j_\sigma}(S')$ and $\exists s_{ij} \in \boldsymbol{b}_i$ subject to $s_{ij} \succ \sigma_{i_\sigma j_\sigma}(S')$; then,

$$s_{ij} \succ \sigma_{i_\sigma j_\sigma}(S') \succ s^{\ominus}_{ij^{\ominus}}$$

From bundle monotonicity (Lemma 1), the assignment of s_{ij} cannot depend on the assignment of $s^{\ominus}_{ij^{\ominus}}$ (because s_{ij} is larger, and thus earlier in the bundle, preventing the value of s_{ij}, depending on the assignment of $s^{\ominus}_{ij^{\ominus}}$); thus, without loss of generality, s_{ij} will not depend on any bids in its own bundle $\boldsymbol{b}_i$ smaller than $\sigma_{i_\sigma j_\sigma}(S')$.

Thus, all that must be shown is that all bids $s_{ij} \succ \sigma_{i_\sigma j_\sigma}(S')$ will not be affected by any of the bids in S'. The bundle construction procedure uses an indicator function

$$h_{ij} \leftarrow \prod_{s_{i'j} \in \mathcal{A}_i} \mathbb{I}(c_{ij} > c_{i'j})$$

(Algorithm 3, line 11). The elements of the indicator function can only be different on tasks j' for which $s'_{i'j'} \in S'$. Therefore, during bundle building, the selection of the next-best bid $s^{\star}_{i^\star j^\star}$ (Algorithm 3, line 14) will return identical results for all bids $s^{\star}_{ij^\star} \succ \sigma_{i_\sigma j_\sigma}(S')$.

A note about Theorem 1 is that it depends on bid warping to ensure Lemma 1. If bundles are not monotonic, then $\exists i$, $s_{ij} \in \boldsymbol{b}_i$, $s^{\ominus}_{ij^{\ominus}} \in \boldsymbol{b}_i$ subject to $s_{ij} \succ s^{\ominus}_{ij^{\ominus}}$ where the assignment of s_{ij} depends on the previous assignment of $s^{\ominus}_{ij^{\ominus}}$ (i.e., the value of the bid on task j increases because of the assignment of task $j^{\ominus}$). This is exactly the condition that can lead to algorithmic cycling, and it is what bid warping prevents.

Definition 4: Define a new bid to be a bid that was not included in the bundle of the previous bundle building iteration.

Theorem 2: The largest new bid $s^{\star}_{ij^\star}$ (Algorithm 3, line14) that can be added to agent i's bundle $\boldsymbol{b}_i$ after receiving a set of overbid messages S' will be smaller than the largest element of S', i.e.,

$$\sigma_{i_\sigma j_\sigma}(S') \succ s^{\star}_{ij^\star} \tag{17}$$

Proof: From Theorem 1, all bids s_{ij} in agent i's bundle $\boldsymbol{b}_i$, subject to $s_{ij} \succ \sigma_{i_\sigma j_\sigma}(S')$, will be preserved. For this proof, construct a new bundle $\boldsymbol{b}_i^{+}$ starting with all old elements $s^{o}_{ij} \in \boldsymbol{b}_i$ subject to $s^{o}_{ij} \succ \sigma_{i_\sigma j_\sigma}(S')$. The strategy for the rest of this proof will be to show that the next bid $s^{\star}_{ij^\star}$ added to $\boldsymbol{b}_i^{+}$ is smaller than $\sigma_{i_\sigma j_\sigma}(S')$. Lemma 1 then guarantees that all other new bids added during the rest of the bundle building operation be smaller than $s^{\star}_{ij^\star}$, and thus less than $\sigma_{i_\sigma j_\sigma}(S')$.

Define $s^{\ominus}_{ij^{\ominus}}$ to be the largest bid in the old bundle $\boldsymbol{b}_i$ that is less than $\sigma_{i_\sigma j_\sigma}(S')$. If no bids in $\boldsymbol{b}_i$ are smaller than $\sigma_{i_\sigma j_\sigma}(S')$, treat $s^{\ominus}_{i^{\ominus}j^{\ominus}}$ as an empty bid of score zero (all bids with positive scores are bigger than it). The rest of this proof will formalize how each overbid $s'_{i'j'} \in S'$ can affect the next largest bid in $\boldsymbol{b}_i^{+}$.

The only way that the next largest bid $s^{\star}_{ij^\star}$ can increase its value compared to its counterpart in the old bundle $s^{\ominus}_{ij^{\ominus}}$ (i.e., $s^{\star}_{ij^\star} \succ s^{\ominus}_{ij^{\ominus}}$) due to receiving an overbid $s'_{i'j'}$ is if an element from the indicator h_{ij} (Algorithm 3 line 11) changes from a zero to a one. This can only occur if $s'_{i'j'}$ is specifically a drop bid, and thus removes a bid on task j' in agent i's local bid space $\mathcal{A}_i$ (Algorithm 6, line 8). This is because only a drop bid can decrease the winning scores in $\mathcal{A}_i$. All other overbids will only increase the winning score in the local bid space. If Algorithm 3, line 11, had previously been returning zero (before the drop bid arrived), it meant that the previous winning value on task j' (before the drop bid) was $c'_{i'j'} > F_{ij'}(\boldsymbol{b}_i^{+})$ (otherwise, $h_{ij'}$ would have returned a one). Therefore, either $s^{\ominus}_{ij^{\ominus}} \succ s'_{i'j'}$ and the next-best bid will remain unchanged ($s^{\star}_{ij^\star} = s^{\ominus}_{ij^{\ominus}}$), and thus $\sigma_{i_\sigma j_\sigma}(S') \succ s^{\ominus}_{ij^{\ominus}} = s^{\star}_{ij^\star}$, or $s'_{i'j'} \succ s^{\ominus}_{ij^{\ominus}}$ and the next largest bid, will become

$$c^{\star}_{ij^\star} = \max\Big(F_{ij'}(\boldsymbol{b}_i^{+}), c^{\ominus}_{ij^{\ominus}}\Big)$$

Therefore, from Definition 1, $\sigma_{i_\sigma j_\sigma} \succeq s'_{i'j'} \succ s^{\star}_{ij^\star}$.

The result shows that no individual overbid $s'_{i'j'}$ can lead to the agent i increasing its next largest bid $s^{\star}_{ij^\star}$ to be larger than $\sigma_{i_\sigma j_\sigma}(S')$. Because $s^{\star}_{ij^\star}$ is simply the largest possible next-best bid, any collection of overbids S' will not allow $s^{\star}_{ij^\star}$ to be larger than $\sigma_{i_\sigma j_\sigma}(S')$ either.

Theorem 3: Every agent of the team using the BW-CBBA agrees on a globally consistent bid space $\mathcal{A}$ in, at most, two $N_t\mathcal{D}$ algorithmic iterations.

Proof: The form of this proof will be to use a virtual agent that can observe the algorithmic progression (without affecting the agents). This observer is able to construct a globally consistent bid space $\mathcal{A}$ by listening to the communication between the agents. The existence of $\mathcal{A}$ can be used to guarantee algorithmic convergence. It will be shown that, once a bid is placed in $\mathcal{A}$, it will never be dropped by the agent that placed the bid or outbid by another agent. Therefore, when all bids made by agents in the team are located in $\mathcal{A}$, the algorithm has converged. The proof will use induction to build up the global bid space $\mathcal{A}$.

Initialize $\mathcal{A} = \{\}$. The base case of the induction proof requires that the first bid added to the global bid space ($|\mathcal{A}| = 1$) will never be dropped by the bidding agent and will never be outbid by any other agent. This is shown by first considering that, for each full plan constructed by the BW-CBBA, the algorithm starts with a bundle building phase (Algorithm 2) where each agent i is initialized with a possibly inconsistent local bid space $\mathcal{A}_i$. If this is the first plan for a new mission, $\mathcal{A}_i$ will be empty, but if this plan is a midmission replan, $\mathcal{A}_i$ will be the last known estimate of the team assignment (which may be outdated). As defined in Algorithm 3, each agent i removes its own bids from its own local bid space $\mathcal{A}_i$ (lines 2–4) and clears its own bundle $\boldsymbol{b}_i = \{\}$ (line 5). After the first iteration of bundle building completes (Algorithm 2, line 11) for all agents i, there are two outcomes relevant to algorithmic convergence:

1) There exists a bid $s'_{i'j'}$ in the local bid space $\mathcal{A}_i$ of some agent i that is larger than all other bids in the network (including all bids in the actual bundle $\boldsymbol{b}_{i'}$ of agent i'). More precisely, $\exists i, s'_{i'j'} \in \mathcal{A}_i$ subject to $\forall\ i, \forall\ s_{ij} \in \boldsymbol{b}_i, s'_{i'j'} \succ s_{ij}$. This arises when one agent is assuming the existence of a large old bid that is no longer valid. When bid information is propagated in the algorithm's consensus phase, agent i will receive a drop bid removing the assignment of $s'_{i'j'}$ (constructed via Algorithm 5 by agent i') in no more than $\mathcal{D}$ algorithmic iterations.

2) After the large outdated bids are removed from bid spaces, there will exist an actual bid $s'_{i'j'}$ in the bundle $\boldsymbol{b}_{i'}$ of some agent i' subject to $\forall\ i$, $\forall\ s_{ij} \in \boldsymbol{b}_i$, s.t.$s_{ij} \neq s'_{i'j'}, s'_{i'j'} \succ s_{ij}$. From Theorem 2, no other agents will be able to generate a larger bid $s^{\star}_{i^{\star}j^{\star}} \succ s'_{i'j'}$ in the future because $s'_{i'j'}$ is the largest bid in the fleet, and thus is the largest bid that can be an element of an overbid set S'. From Theorem 1, $s'_{i'j'}$ will remain in the bundle of agent i' forever; because no other agent can outbid $s'_{i'j'}$, it will never receive an overbid larger than $s'_{i'j'}$. After $\mathcal{D}$ consensus phases, all agents will receive $s'_{i'j'}$; thus, overall, in, at most, $2\mathcal{D}$ algorithmic iterations, $s'_{i'j'}$ can be added to $\mathcal{A}$ and $|\mathcal{A}| = 1$.

At the end of this base case, assume that all outdated bids from a previous planning iteration are removed from the entire fleet, which would have taken, at most, $\mathcal{D}$ iterations to remove.

Therefore, it must be shown that, if we assume a global bid space $\mathcal{A}$ of size $|\mathcal{A}| = n$ and that all bids currently in $\mathcal{A}$ will never be dropped or outbid, then in, at most, $2\mathcal{D}$ algorithmic iterations, there either exists another bid to add to $\mathcal{A}$ or the algorithm has converged. More formally, either $\forall\ i \nexists s_{ij} \in \boldsymbol{b}_i$ subject to $s_{ij} \in \mathcal{A}$ (the algorithm has converged) or $\exists s'_{i'j'} \in \boldsymbol{b}_{i'}, s'_{i'j'} \notin \mathcal{A}$ subject to $\forall\ i, \forall\ s_{ij} \in \boldsymbol{b}_i, s_{ij} \notin \mathcal{A}, s'_{i'j'} \neq s_{ij}, s'_{i'j'} \succ s_{ij}$.

In this step, the following three scenarios can arise:

1) There are no bids in the network that are not already in $\mathcal{A}$ or. more formally, $\forall\ i \nexists s_{ij} \in \boldsymbol{b}_i$ subject to $s_{ij} \notin \mathcal{A}$. In this case, the algorithm has converged.

2) A bid s_{ij} has been created and has been inserted into the global bid space $\mathcal{A}$ but agent i' has yet to receive a message containing s_{ij}, and thus has a bid on task j in its bundle $\boldsymbol{b}_{i'}$ that is smaller than s_{ij}. More formally, the scenario arises if $\exists s_{ij} \in \mathcal{A}$ subject to $\exists s'_{i'j} \in \boldsymbol{b}_{i'}$ subject to $i \neq i'$. By the inductive assumption, s_{ij} will never be outbid. Therefore, in (at most) $\mathcal{D} - 1$ iterations, i' will receive a message about s_{ij} and drop its bid $s'_{i'j}$. When $s'_{i'j}$ is dropped, agent i' may be forced to drop other tasks as well that were dependent on the assignment of $s'_{i'j}$. Define the largest bid of these additional dropped bids as $s^{\ominus}_{i'j^{\ominus}}$. If the drop bid $s^{\ominus}_{i'j^{\ominus}}$ is larger than all other bids not yet in $\mathcal{A}$ (if $\forall\ i$, $\forall\ s_{ij} \in \boldsymbol{b}_i, s_{ij} \notin \mathcal{A}, s^{\ominus}_{i'j^{\ominus}} \neq s_{ij}$, $s^{\ominus}_{i'j^{\ominus}} \succ s_{ij}$), then the algorithm requires an additional $\mathcal{D}$ iterations to allow for this drop bid to propagate to all agents. This is required because $s^{\ominus}_{i'j^{\ominus}}$ will be in the overbid set S' that is being communicated to all agents and, according to Theorem 2, bids can be added up to the size of $s^{\ominus}_{i'j^{\ominus}}$. Thus, a new largest bid will be possible until all agents have received a drop bid message of $s^{\ominus}_{i'j^{\ominus}}$. This results in $2\mathcal{D} - 1$ algorithmic iterations and a transition into the criteria for scenario 3 as follows.

3) There exists a bid $s'_{i'j'}$ in some agent i''s bundle $\boldsymbol{b}_{i'}$ that is larger than all other bids s_{ij} in every other agent's bid spaces that is not currently located in $\mathcal{A}$ ($\exists s'_{i'j'} \in \boldsymbol{b}_{i'}$ subject to $s'_{i'j'} \notin \mathcal{A}$ and $\forall\ i$, $\forall\ s_{ij} \in \mathcal{A}_i$, $\forall\ s_{ij} \neq s'_{i'j'}, s_{ij} \notin \mathcal{A}, s'_{i'j'} \succ s_{ij}$). Because no tasks in $\mathcal{A}$ can be outbid (inductive assumption) and no agents are currently outbid on tasks in $\mathcal{A}$ (which is handled by the aforementioned scenario 2), when the largest bid in the team not yet in $\mathcal{A}$ ($s'_{i'j'}$) is shared, no other agents will be able to outbid it (due to Theorem 1). Additionally, because i' will never receive an outbid message greater than $s'_{i'j'}$ (because no other agents can bid higher (Theorem 2), $s'_{i'j'}$ will stay in the bundle of agent i forever (Theorem 1). Therefore, $\mathcal{A} \leftarrow \mathcal{A} \cup s'_{i'j'} |\mathcal{A}| = n + 1$ in one iteration.

Therefore, incrementing $\mathcal{A}$ can be achieved in, at most, $2\mathcal{D}$ algorithmic iterations. This then proves that a globally consistent bid space can be constructed during algorithmic execution in, at most, $2N_t\mathcal{D}$ algorithmic iterations (two $\mathcal{D} - 1$ iterations from scenario 2 and one iteration from scenario 3 for each task). This also means that every individual agent will agree on the full bid space in, at most, $2N_t\mathcal{D}$; thus, BW-CBBA has converged.

B. Performance Guarantee

This section will define when nontrivial performance guarantees for BW-CBBA are available and how close to optimal these guarantees are. The form of the following performance analysis is inspired by theorem 11 in [35]. This section will use set function notation when referring to objective functions. Therefore, the notation $F(\mathcal{A})$ will specify the score function F evaluated on the bid space $\mathcal{A}$. Furthermore, define the notation $\mathcal{A}^{\mathrm{BW}}_F$ to be the bid space returned using objective function F with the BW-CBBA. Similarly, define $\mathcal{A}^{\star}_F$ to be the optimal bid space with respect to objective function F. Additionally, define $\mathcal{A}_k$ to be a bid space constructed from the largest k warped bids of $\mathcal{A}^{\mathrm{BW}}_F$ where $s_{i_k j_k}$ is defined to be the kth element added to $\mathcal{A}^{\mathrm{BW}}_F$ (which was constructed by agent i_k on task j_k) using the global bid space construction procedure from Theorem 3.

The form for the following proof will be to compare the optimal allocation $\mathcal{A}^{\star}_F$ evaluated on the desired non-submodular objective function $F(\mathcal{A}^{\star}_F)$ to the optimal allocation $\mathcal{A}^{\star}_{F_k}$ over a sequential set of modified objective functions F_k. Before describing the main result, a few definitions and a lemma will be needed.

Definition 5: Define the sequence of score functions F_k over a subset $\mathcal{J}_k$ of the full task set $\mathcal{J}$ where $\mathcal{J}_k \leftarrow \mathcal{J} \setminus \bigcup_{l=1}^{k} j_k$ as

$$F_0(\mathcal{A}) = F(\mathcal{A})$$
$$F_k(\mathcal{A}) = F_{k-1}(\mathcal{A} \cup s_{i_k j_k}) - F_{k-1}(s_{i_k j_k}), \quad \forall\ k \in [|\mathcal{J}|]$$

where $[|\mathcal{J}|]$ is defined as $\{1, \ldots, |\mathcal{J}|\}$. This can be equivalently defined using partial bid spaces and the original objective function F as

$$F_k(\mathcal{A}) = F(\mathcal{A} \cup \mathcal{A}_k) - F(\mathcal{A}_k), \quad \forall\ k \in [|\mathcal{J}|]$$

Both forms of F_k will be used for proving convergence guarantees.

Definition 6: Define a submodular lower bound function H to the desired mission objective function F as

$$H(\varnothing) = F(\varnothing)$$
$$H(\mathcal{A} \cup s) - H(\mathcal{A}) = \min_{\mathcal{A}' \subseteq \mathcal{A}} (F(\mathcal{A}' \cup s) - F(\mathcal{A}')) \quad \forall\ s, \mathcal{A}$$

Definition 7: Define a parameter ϵ that defines a measure on the non-submodularity of the mission objective function F as

$$1 + \epsilon = \max_{s,\mathcal{A}} \frac{F(\mathcal{A} \cup s) - F(\mathcal{A})}{H(\mathcal{A} \cup s) - H(\mathcal{A})}$$

Given that ratios are used in the definition of ϵ, it is important that F is monotonic. Other alternative measures of non-submodularity are available and would be needed for nonmonotonic functions. If the desired mission objective function F is actually submodular, then $\epsilon = 0$ and $F = H$.

Combining Definitions 7 and 6 provides a submodular upper and lower bound on F for all bid spaces $\mathcal{A}$ as

$$(1 + \epsilon)H(\mathcal{A}) \geq F(\mathcal{A}) \geq H(\mathcal{A}), \quad \forall\ \mathcal{A} \tag{18}$$

Definition 8: Define the minimum possible bid $c_k^{\min}$ on task j_k as

$$c_k^{\min} = \min_{\mathcal{A}} F(\mathcal{A} \cup s_k) - F(\mathcal{A})$$

where, for monotonic functions, $c_k^{\min} > 0$.

Lemma 2: The scores $\bar{c}_{i_k j_k}$ of warped bids $\bar{s}_{i_k j_k}$ will be greater than or equal to the bid's incremental value evaluated on objective function H as defined in Definition 6:

$$\bar{c}_{i_k j_k} \geq H(\mathcal{A}_{k-1} \cup s_{i_k j_k}) - H(\mathcal{A}_{k-1})$$

Proof: Two cases of either of the following cases are possible:

1) When warped, $s_{i_k j_k}$ does not change its value; thus, its warped value is the incremental score with respect to objective function F

$$\bar{c}_{i_k j_k} = F(\mathcal{A}_{k-1} \cup s_{i_k j_k}) - F(\mathcal{A}_{k-1}) \overset{\text{Def.6}}{\geq} H(\mathcal{A}_{k-1} \cup s_{i_k j_k}) - H(\mathcal{A}_{k-1})$$

2) The value $\bar{c}_{i_k j_k}$ of the warped bid $\bar{s}_{i_k j_k}$ has decreased due to bid warping. This requires that at least one bid constructed by agent i_k is already in $\mathcal{A}_{k-1}$. These bids already in $\mathcal{A}_{k-1}$ will be called $\mathcal{A}^{\ominus}$, where $\mathcal{A}^{\ominus} \subseteq \mathcal{A}_{k-1}$. Define bid $\bar{s}_{i_k j_{k^{\ominus}}} \in \mathcal{A}^{\ominus}$ to be the winning bid at iteration $k^{\ominus}$. The values of the unwarped bids located in $\mathcal{A}^{\ominus}$ were defined as

$$c_{i_k j_{k^{\ominus}}} = F(\mathcal{A}_{k^{\ominus}-1} \cup s_{i_k j_{k^{\ominus}}}) - F(\mathcal{A}_{k^{\ominus}-1}) \tag{19}$$

At iteration $k^{\ominus}$, the bid on task $j_{k^{\ominus}}$ by agent i_k was the winning bid; therefore, at that time, it was greater than agent i_k's bid on task j_k:

$$c_{i_k j_{k^{\ominus}}} \geq F(\mathcal{A}_{k^{\ominus}-1} \cup s_{i_k j_k}) - F(\mathcal{A}_{k^{\ominus}-1}), \quad \forall\ k^{\ominus} \tag{20}$$

From the definition of bid warping in Eq. (11) and the assumption of this proof clause that the value of $\bar{s}_{i_k j_k}$ has decreased due to bid warping, the warped bid on task j_k is defined in terms of a minimum over all of the bids located in agent i_k's current bundle (which has the same elements as $\mathcal{A}^{\ominus}$):

$$\bar{c}_{i_k j_k} = \min_{s_{i_k j_{k^{\ominus}}} \in \mathcal{A}^{\ominus}} c_{i_k j_{k^{\ominus}}} \tag{21}$$

and therefore, from the definition of H (Definition 6), a minimum over a larger set will always be smaller; thus,

$$\bar{c}_{i_k j_k} \geq H(\mathcal{A}_{k-1} \cup s_{i_k j_k}) - H(\mathcal{A}_{k-1}) \tag{22}$$

Theorem 4: If there exists an H as defined in Definition 6 and ε as defined in Definition 7, and the mission objective function F is monotonic, then a provable performance bound between the BW-CBBA allocation $\mathcal{A}_F^{\mathrm{BW}}$ and the optimal allocation $\mathcal{A}_F^{\star}$ exists as

$$F(\mathcal{A}_F^{\star}) \leq (2 + \varepsilon)F(\mathcal{A}_F^{\mathrm{BW}}) \tag{23}$$

Proof: To simplify the notation of the following proof, a few notational substitutions will be made:

1) Note that $s_k^{\star} \leftarrow s_{i_k^{\star} j_k}^{F_{k-1}}$, where $s_{i_k^{\star} j_k}^{F_{k-1}}$ is the bid made on task j_k in the optimal assignment using objective function F_{k-1}

2) Note that $s_k \leftarrow s_{i_k j_k}$ to represent the kth bid from the BW-CBBA bid space $\mathcal{A}_F^{\mathrm{BW}}$.

3) $\mathcal{A}_{\ominus}^{\star} \leftarrow \mathcal{A}_{F_{k-1}}^{\star} \setminus s_k^{\star}$ represents the optimal bid space with regard to F_{k-1} without the bid $s_k^{\star}$ that was made on task j_k.

Begin the proof by noting that the value of the optimal bid space constructed with regard to objective function F_k will always be greater than or equal to the evaluation of any other bid space on this objective function F_k, so that

$$F_k(\mathcal{A}^\star_{F_k}) \geq F_k(\mathcal{A}^\star_\ominus) \tag{24}$$

Then, using the definition of F_k from Definition 5, the right-hand side can be expanded as

$$F_k(\mathcal{A}^\star_\ominus) = F_{k-1}(\mathcal{A}^\star_\ominus \cup s_k) - F_{k-1}(s_k)$$

which can then be expanded as three terms:

$$F_k(\mathcal{A}^\star_\ominus) = F_{k-1}(\mathcal{A}^\star_{F_{k-1}}) - \tag{25a}$$

$$\Big(F_{k-1}(\mathcal{A}^\star_{F_{k-1}}) - F_{k-1}(\mathcal{A}^\star_\ominus)\Big) - \tag{25b}$$

$$\Big(F_{k-1}(s_k) - \Big(F_{k-1}(\mathcal{A}^\star_\ominus \cup s_k) - F_{k-1}(\mathcal{A}^\star_\ominus)\Big)\Big) \tag{25c}$$

The three terms correspond to the value of the optimal allocation over F_{k-1} [Eq. (25a)], the incremental value for the bid on task j_k in the optimal allocation using objective function F_{k-1} [Eq. (25b)], and the change in the value of the bid made in the bid warped allocation s_k due to the addition of the optimal bid space with regard to F_{k-1} without the optimal bid on task j_k (which was previously defined as $\mathcal{A}^\star_\ominus$) [Eq. (25c)].

The next step is to further rewrite the term defined as Eq. (25b). First observe that $\bar{s}_{i_k j_k}$ was the kth element added to the global bid space during the BW-CBBA so, conditional on an already locked-in bundle of $\mathcal{A}_{k-1}$,

$$\bar{c}_{i^\star_k j_k} \leq \bar{c}_{i_k j_k} \leq c_k \tag{26}$$

where $\bar{c}_{i^\star_k j_k}$ is the warped value that agent $i^\star$ (the optimal winner in allocation $\mathcal{A}^\star_{F_{k-1}}$) could have bid given a bid space of $\mathcal{A}_{k-1}$, $\bar{c}_{i_k j_k}$ is the actual warped value bid in the BW-CBBA, and c_k is the unwarped value of that bid. The relation $\bar{c}_{i^\star_k j_k} \leq \bar{c}_{i_k j_k}$ holds because, if it did not, bid $\bar{s}_{i^\star_k j_k}$ would have been chosen instead as the kth element in the bid warped allocation. Additionally, $\bar{c}_{i_k j_k} \leq c_k$ because bid warping can only decrease the value of a bid. From Lemma 2,

$$\bar{c}_{i^\star_k j_k} \geq H(\mathcal{A}_{k-1} \cup s_{i^\star_k j_k}) - H(\mathcal{A}_{k-1}) \tag{27}$$

the potential warped candidate bid $\bar{s}_{i^\star_k j_k}$ created by agent $i^\star_k$ will be greater than its incremental contribution defined over H. Because H is submodular, adding the rest of the optimal assignments from $\mathcal{A}^\star_{F_{k-1}}$ to the evaluation of the incremental value of $s_{i^\star_k j_k}$ can only decrease its value:

$$\bar{c}_{i^\star_k j_k} \geq H(\mathcal{A}_{k-1} \cup \mathcal{A}^\star_{F_{k-1}}) - H(\mathcal{A}_{k-1} \cup \mathcal{A}^\star_{F_{k-1}} \setminus s_{i^\star_k j_k}) \tag{28}$$

Incorporating this with the definitions of H (Definition 6) and ϵ (Definition 7) from Eq. (18) and the equivalence of the two forms of F_k from Definition 5 implies that

$$(1+\epsilon)\bar{c}_{i^\star_k j_k} \geq \Big(F_{k-1}(\mathcal{A}^\star_{F_{k-1}}) - F_{k-1}(\mathcal{A}^\star_\ominus)\Big) \tag{29}$$

Finally, combining Eq. (29) with Eq. (26) provides a bound on the term defined as Eq. (25b):

$$(1+\epsilon)c_k \geq \Big(F_{k-1}(\mathcal{A}^\star_{F_{k-1}}) - F_{k-1}(\mathcal{A}^\star_\ominus)\Big) \tag{30}$$

The next objective is to bound the term defined as Eq. (25c). From the definition provided as Definition 8, Eq. (25c) can be upper bounded as $c_k - c_k^{\min}$. If this result and Eq. (30) are substituted into Eqs. (25b) and (25c) and sequentially iterated for all k using the relation in Eq. (24), an optimal performance bound can be achieved as

$$F(\mathcal{A}^\star_F) \leq \sum_{k=1}^{|\mathcal{J}|}((2+\epsilon)c_k - c_k^{\min}) \tag{31}$$

If functions are only known to be at least monotonic ($c_k^{\min} = 0, \forall\ k$), this can be simplified to the desired result:

$$F(\mathcal{A}^\star_F) \leq (2+\epsilon)F(\mathcal{A}^{\mathrm{BW}}_F) \tag{32}$$

C. Handling Nondeterministic Score Functions

As was mentioned previously in Sec. IV.C, it is necessary to have repeatable evaluations of score functions. This constraint allows the use of stochastic score functions as long as the approximate evaluation of stochastic metrics is repeatable. For example, if the evaluation technique uses the same set of particles for sampling the uncertainty, or the same algorithm seed at every evaluation of the score function, then the function will return a repeatable value. Truly stochastic function evaluations (where their value is not repeatable), however, do not have absolute convergence guarantees with this approach. Certain stochastic distributions may converge almost surely, but these special cases are not further explored in this paper.

VI. Results Using BW-CBBA

This section provides performance and convergence comparisons between several global information consistency assumption algorithms and local information consistency assumption algorithms. The results will show that LICA algorithms can significantly reduce algorithmic convergence time over competing GICA algorithms. Additionally, the bid warping approach described in Sec. IV.B can significantly improve the performance of LICA algorithms. In fact, in all of the domains tested for this paper, the BW-CBBA actually returns the same allocation as comparable GICA algorithms.

A. Non-Submodular Fuel Penalty: Two-Agent Case

The first example considers a simple mission where two agents achieve reward by visiting a set of locations in the environment. The score function associated with this mission is defined as follows:

$$J = \sum_{i=1}^{N_a}\left(\sum_{j=1}^{N_t} R x_{ij}\right) - f_i d_i(\boldsymbol{b}_i) \tag{33}$$

where a reward of R is obtained for each task visited, and cost is defined as the fuel cost f_i multiplied by the distance travelled $d_i(\boldsymbol{b}_i)$ by agent for its assigned group of tasks $\boldsymbol{b}_i$. Figure 6 visually compares the planned paths for a two-agent 30-task mission using the original baseline CBBA with a submodular approximate score function (Fig. 6a) and the BW-CBBA augmented to use the true non-submodular score function (Fig. 6b). The numerical values used for this experiment were a reward of $R = 100$ and a fuel penalty of $f_i = 10$. As was introduced in example 3 of Sec. III, one heuristic approach to ensure submodularity within the original CBBA framework involves approximating the cost in Eq. (33) by a distance measure based only on the initial agent position and the task locations. This heuristic score function cannot explicitly capture how task desirability can increase due to the assignment of other tasks. This results in the algorithm's selection criteria being driven by the tasks proximity to the agent's initial position instead of where it will fit into the agent's current path (Fig. 6a). Conversely, Fig. 6b demonstrates how the BW-CBBA uses the non-submodular objective function to create intuitively much better assignments by capturing the inherent non-submodularity in the desired objective function.

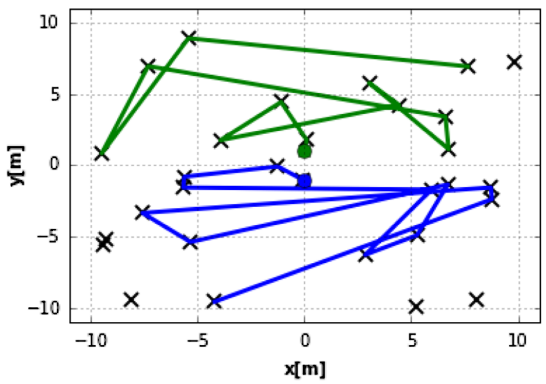
B. Non-Submodular Fuel Penalty: Monte Carlo Results

This experiment provides Monte Carlo results comparing the performance in various mission scenarios for five different algorithms. The scenarios used in this section were designed to show two things:

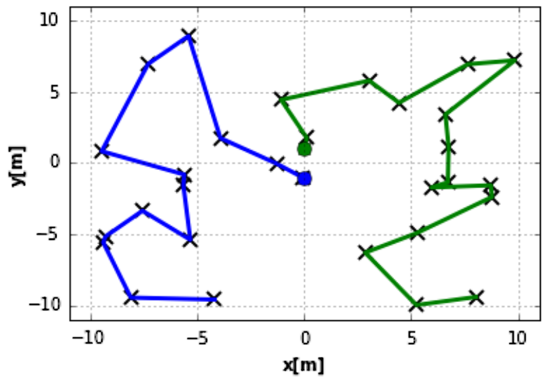
1) Even in environments where reaching global consistency is possible, GICA algorithms can require many iterations to reach a convergent solution.

2) By using the BW-CBBA, the score performance gap of using a LICA algorithm is negligible in practice.

The environment for these Monte Carlo tests places tasks and agents at random locations in a two-dimensional rectangle of dimensions of roughly 34 by 12. (These numbers correspond to the shape of the laboratory physical flight volume.) The agents are modeled in a continuous domain where the speed of the agents was fixed at a maximum of 0.6, the time agents were required to pause to "complete" the tasks was set to one time unit, and all tasks expired after 100 time units. The objective function for the environment was identical to Eq. (33) with a reward of $R = 100$ and a fuel penalty of $f_i = 10$. The agents have identical fuel penalties and speeds, but this is not required by any algorithms used in this

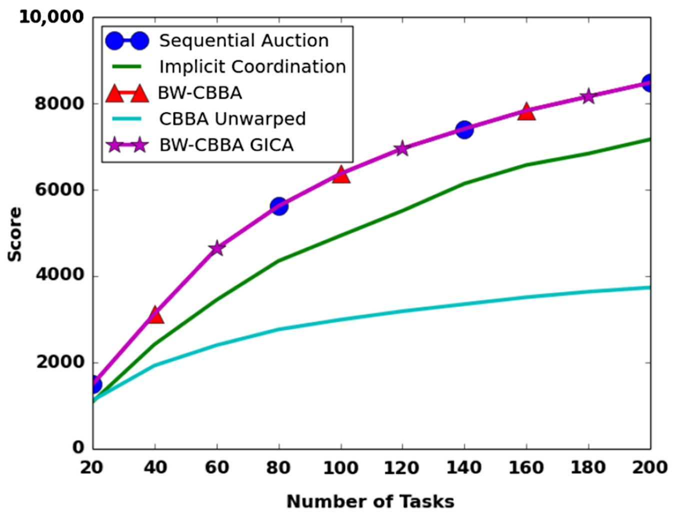


a) CBBA

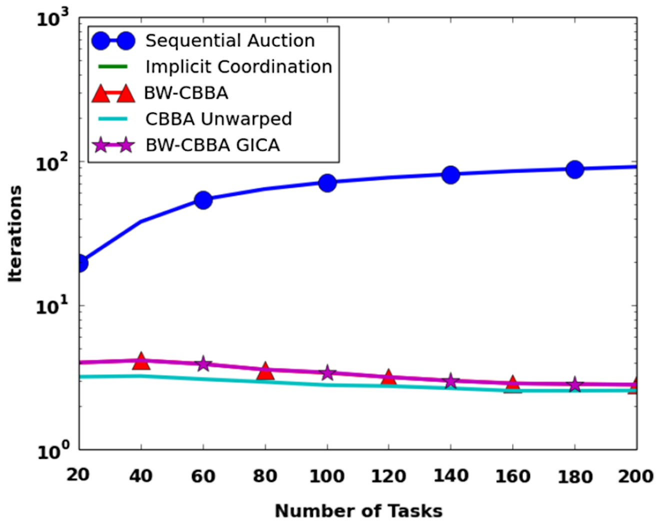


b) BW-CBBA

Fig. 6 Comparison of planned paths for a two-agent 30-task mission: a) original CBBA with a submodular heuristic score function, and b) BW-CBBA.

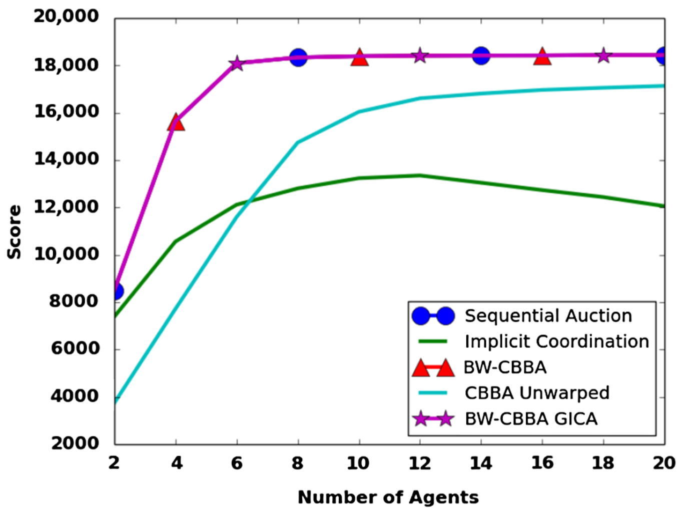


a) Two-agent example (score)

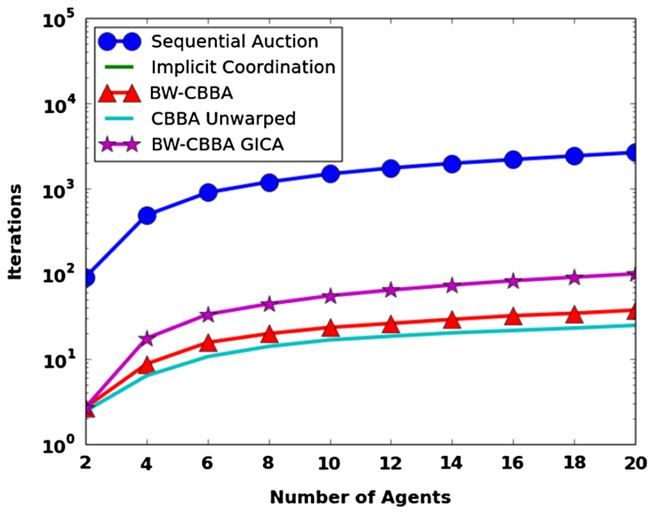


b) Two-agent example (iterations)

Fig. 7 Results for two-agent Monte Carlo run.



a) 200-task example (score)



b) 200-task example (iterations)

Fig. 8 Results for 200-task Monte Carlo run.

experiment. The agents' objective functions, however, are heterogeneous because they depend on both the agents' starting locations and the full bundles assigned to each agent. Every data point in Figs. 7 and 8 is an average over 500 Monte Carlo trials. The computational environment for these experiments used a separate computer for each agent communicating over a virtual communication network, which was defined as a minimum spanning tree over a distance metric corresponding to the agents' randomized starting positions. This provided a level of realism to the simulated performance of the algorithms because the only information shared between the agents was the actual task bids, and thus true communication synchronization was required between the distributed agents. The five planners used in these tests were as follows:

1) The first planner is Sequential Auction, which is a GICA algorithm (as defined in [22]) that essentially involves the team incrementally building up a global bid space one task at a time. This requires every member of the team communicating with all other agents for every single task assignment.

2) The second planner is Implicit Coordination. The implicit coordination implementation in this paper uses a poor information environment because each agent independently optimizes its own objective function, ignoring the contributions of other agents. The result gives the teamwide performance when no explicit cooperation is used.

3) The BW-CBBA (as defined in Sec. IV.B) is the LICA contribution of this paper.

4) CBBA unwarped refers to the baseline solution described in [26], which uses an a priori approximate submodular function.

5) The BW-CBBA with GICA, which is a variant of the BW-CBBA implemented for this paper, ensures that every agent's local bid space is equivalent to the global bid space before every bundle building phase. This algorithm is essentially a bundle version of a sequential auction, and thus approximates the fastest expected convergence time of a GICA auction algorithm.

Planners not run in this test include those that predict assignments for teammates and incorporate this information into their prospective assignments. These algorithms require sharing a different domain of information (including information about other agents' actual objective functions) and are not considered in this paper. Traditionally, these approaches would be considered GICA algorithms because they require information about the entire fleet to guarantee convergence [5,7,36], but recent work has looked at LICA algorithms that use information about other agents' score functions [37].

The experiments shown in Figs. 7a and 7b are two-agent Monte Carlo runs with 500 trials averaged for each data point while varying the number of tasks. Figure 7a shows that the BW-CBBA performs identically in score to both of the GICA algorithms. The implicit coordination technique performs reasonably well because the tasks expired at 100 time units, which made it impossible for both agents to service all of the tasks. Thus, there was relatively little overlap in desired assignments between the agents in this no-coordination planner. The unwarped CBBA actually performed quite poorly, especially for high numbers of tasks, because its objective function could not capture the supermodular coupling between servicing tasks that were near each other. Figure 7b highlights the convergence times for each of these algorithms. The first remarkable aspect about this figure is the large number of iterations required for convergence using the sequential auction algorithm. Even though there are only two agents, and thus the network diameter is one, it takes a full iteration for the assignment of every single task. The BW-CBBA and BW-CBBA GICA use significantly fewer iterations for all sized task environments and actually require the same number of iterations to reach convergence. This is because the two-agent network is fully connected so that the agents using the BW-CBBA can have consistent local bid spaces after every iteration. Thus, the algorithmic execution of the BW-CBBA and BW-CBBA GICA is identical. These approaches take less than one more iteration on average than the unwarped CBBA. The extra convergence time is due to a slightly more complicated optimization between the two agents, which is taking into account the non-submodular objective function. Because there is no explicit coordination with implicit coordination, it always "converges" in one iteration. The takeaway from Figs. 7a and 7b is that, even with two agents, the BW-CBBA outperforms the unwarped CBBA significantly in score performance for a small penalty in an increased number of iterations for convergence. Additionally, its performance is identical to the tested GICA planners.

The experiments shown in Figs. 8a and 8b are Monte Carlo runs with 200 task environments and 500 trials averaged to create each data point while varying the number of agents. Figure 8a shows identical performance between the BW-CBBA and the GICA algorithms. Additionally, it shows a significant performance gap between the unwarped CBBA and the implicit coordination approach, especially for smaller team sizes. When the team size reaches eight agents and above, the BW-CBBA and the GICA algorithms are able to service all of the tasks efficiently under the 100 time unit task deadlines (the only improvement comes from agents potentially starting nearer to desired tasks). The original unwarped CBBA, even with 20 agents, still has a mission performance gap because the objective function is unable to capture the inherent coupling of travel distance in the objective functions. Theoretically, this performance gap may exist until each agent is only servicing a single task, in which case the unwarped CBBA and BW-CBBA will return the same allocation. The performance of implicit coordination actually degrades after 12 agents because the costs of overlapping assignments start outweighing the benefits of having more agents to service difficult-to-reach tasks. Figure 8b highlights the number of iterations each planner requires to reach convergence. Again, the sequential auction GICA algorithm takes significantly more iterations to converge than any of the other planners. In fact, for the most difficult assignment problems of 20 agents and 200 tasks, this algorithm was requiring nearly 2000 iterations on average. This is due to the fact that 200 tasks are assigned incrementally across an average network diameter of 10. The BW-CBBA GICA performs significantly better than the sequential auction because the coupling in the problem allows agents to agree on many task assignment winners at the same time. Assigning multiple tasks simultaneously allows this auction to reduce the convergence time by more than an order of magnitude. The BW-CBBA, which is a LICA algorithm, further reduces the number of iterations to convergence below the BW-CBBA GICA. In fact, for the largest problem sizes shown, the BW-CBBA converges 50 iterations sooner. Intuitively, this is because agents will rarely have allocation conflicts with teammates that are highly separated across the communication network. Therefore, conflict resolution is more efficiently conducted using local communication. The CBBA unwarped converges marginally faster than even the BW-CBBA but, again, this is at the expense of significant degradation in score performance. The takeaway from Figs. 7 and 8 is that the BW-CBBA significantly improves the performance of traditional LICA algorithms (unwarped CBBA) while converging in significantly fewer iterations than GICA algorithms. There is a hidden computation cost not shown in Figs. 7 and 8 that involves the onboard agent computation of the desired assignments. For some domains, this extra computation may be quite relevant to the convergence times of the algorithms and is worth investigation, but this paper is focused on understanding the information assumptions and communication costs. Despite this, for the tests shown in Figs. 7 and 8, agent computation times were negligible compared to the required infrastructure to synchronize communication between the decentralized agents.

C. Remark

It should be noted that the approach in this paper has focused specifically on what are called task consensus algorithms that utilize consensus protocols that share assignment information between agents. An introduction to how these algorithms compare to a algorithms that share state information between themselves (called implicit coordination algorithms) is introduced in [38]. Specifically, task consensus algorithms are good in loosely coupled domains and can have strong guarantees on performance and convergence. However, since they share assignments, they can have long convergence times when tasks have highly-coupled assignment constraints between each other (the details of this are extensively discussed in [38]). In these domains, implicit coordination algorithms can do a good job with these highly coupled assignments, but again at the downside that they often are unable to make any guarantees on performance. Other work presented in chapters 4 and 5 of [39] leverages the understanding of LICA algorithms introduced in this paper and implicit information about other agents to efficiently produce provably good assignments in environments that require coupled assignments.

VII. Conclusions

Submodularity is a powerful property that can be exploited for provable performance and convergence guarantees in distributed task allocation algorithms. However, some mission scenarios cannot easily be approximated as submodular a priori. This paper introduces an algorithmic extension for LICA algorithms that enables them to converge using non-submodular score functions. These enhancements use non-submodular ranking of tasks within each agent's internal decision making while externally enforcing that shared bids appear as if they are created using submodular score functions. Convergence and performance bounds are proven for this new algorithm called the BW-CBBA. The numerical results of this effort show significant improvements over hand-tuned heuristic approaches that approximate the true non-submodular score functions.

Appendix: Analysis of Algorithmic Performance

Definitions 6 and 7 provide theoretical insight into the tightness of the performance bounds provided in Theorem 4. This Appendix provides some practical insight into how the BW-CBBA works in practice.

An optimal upper bound for the score function defined as Eq. (33) in Sec. VI can be defined as

$$\sum_{j=1}^{N_t} \max\Big(\max_i (R_{ij} - f_i d_{ij}), \max_{i,j'} (R_{ij} - f_i d_{j'j}), 0\Big) \tag{A1}$$

where d_{ij} is the distance from agent i to task j. This bound essentially finds the maximum possible reward achievable by each task, without the constraints of connected trajectories by agents. This upper bound will almost always be loose, but nevertheless will give some insight into the performance of the algorithms in Sec. VI.B.

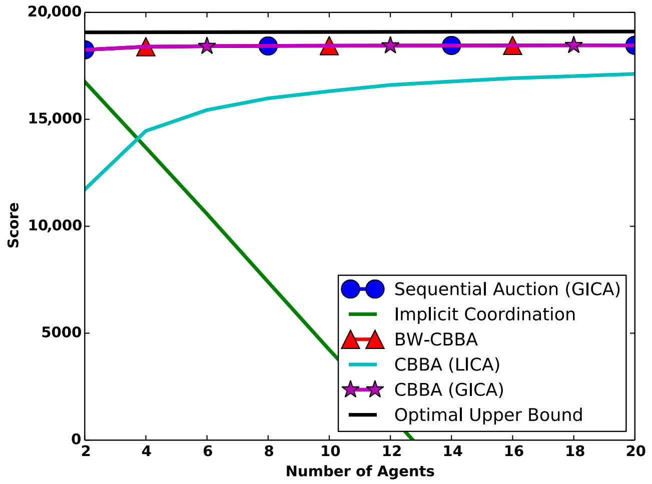


Fig. A1 Comparison of mission score in 500 Monte Carlo trials of a 200-task mission with a varying number of agents.

1) The first category is approximately submodular. With these objective functions, there will be very little difference between F and H; therefore, there will be relatively consistent bounds with respect to optimal. Bid warping will have little effect on the final solution. The only purpose it will serve is to slightly augment the scores when needed to guarantee convergence. An example of this type of score function is in Eq. (33) when f_i is small. In this case, the true objective function is very nearly modular and the final solution will be near optimal.

2) Non-submodular with good local optima is the category that the simulations in Sec. VI demonstrate. The objective function used is not close to submodular because the fuel penalty can span a wide range of values from near zero (when an existing trajectory passes through a previously unassigned task) to multiple times the value of the task (for tasks far away from the iterative assignment's current trajectory). This means that it will be quite difficult to construct a good a priori submodular approximation. The results presented in Figs. 7 and 8 were specifically designed to highlight how the BW-CBBA could perform well in substantially non-submodular environments; as a result, there does not exist a nontrivial ϵ and H for the objective function defined as Eq. (33). Figure A1 presents a slight modification to the problem statement presented in Sec. VI.B. To create a reasonable optimal upper bound, the problem was simplified by removing the task timeout constraints. As can be seen from the optimal upper bound in Fig. A1, all of the GICA approaches and the BW-CBBA perform well (as the uppermost line is an upper bound on optimal and not necessarily tight). The BW-CBBA has good performance in environments where any centralized greedy algorithms can produce good solutions. These environments occur when greedy allocations do not catastrophically degrade the teamwide performance. This occurs in domains like those presented in Sec. VI.B, where tasks are placed randomly in a two-dimensional grid; therefore, it would be difficult to be in a situation where the agent geometry prevents servicing large portions of the environment. It would take a very specific malicious environment for a greedy allocation to perform poorly.

3) The third category is non-submodular with poor local optima. If there are tasks that can be greedily chosen by agents that prevent those agents from servicing other requirements, then the BW-CBBA can perform arbitrarily poorly. In these domains, there can be sufficient objective function coupling such that no sequential greedy algorithm can return a good solution, regardless of the information assumptions. An example of this type of environment is when no agents are close enough to any task to create an initial positive score. Therefore, starting with an empty bid space, the marginal score for adding any task will be negative and the final allocation will be empty. An optimal allocation in this scenario could foreseeably take a negative incremental score on some initial tasks in order to achieve a larger positive reward for other important tasks. Physically, this could be realized if all of the agents start in one corner of the environment and all of the tasks are on the other side of the space. This is a problem with sequential task assignment, and not just the BW-CBBA, and is in general a very difficult problem to solve, even with centralized methods. Indeed, all sequential assignment algorithms will be a poor choice for these environments, and more advanced approaches that evaluate bundles of assignments simultaneously will be needed. In general, these problems are very computationally hard (specifically, NP hard) but, in special cases, other approximate solvers may be able to explore these complex spaces efficiently.

Acknowledgments

This work was sponsored (in part) by the U.S. Air Force Office of Scientific Research and the U.S. Air Force under grant (FA9550-11-1-0134). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Air Force Office of Scientific Research or the U.S. Government. The second author gratefully acknowledges support by the Information and Communication Technology (ICT) Research & Development program of Ministry of Science, ICT, and Future Planning & Institute for Information and communication Technology Promotion (R-20150223-000167, Development of High-Reliability Communications and Security Software for Heterogeneous Unmanned Vehicles).

References

[1] Ponda, S. S., Johnson, L. B., Geramifard, A., and How, J. P., "Cooperative Mission Planning for Multi-UAV Teams," *Handbook of Unmanned Aerial Vehicles*, Springer, New York, 2012, pp. 1448–1484.

[2] 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, No. 9, 2004, pp. 939–954.
doi:10.1177/0278364904045564

[3] Dias, M. B., Zlot, R., Kalra, N., and Stentz, A., "Market-Based Multirobot Coordination: A Survey and Analysis," *Proceedings of the IEEE*, Vol. 94, No. 7, 2006, pp. 1257–1270.
doi:10.1109/JPROC.2006.876939

[4] Bertsekas, D. P., and Tsitsiklis, J. N., *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, Belmont, MA, 1997, pp. 2–3.

[5] McLain, T. M., and Beard, R. W., "Coordination Variables, Coordination Functions, and Cooperative Timing Missions," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 1, 2005, pp. 150–161.
doi:10.2514/1.5791

[6] Castanon, D. A., and Wu, C., "Distributed Algorithms for Dynamic Reassignment," *IEEE Conference on Decision and Control (CDC)*, Vol. 1, IEEE, Piscataway, NJ, Dec. 2003, pp. 13–18.

[7] Curtis, J., and Murphey, R., "Simultaneous Area Search and Task Assignment for a Team of Cooperative Agents," *AIAA Guidance, Navigation, and Control Conference (GNC)*, AIAA Paper 2003-5584, 2003.

[8] Ren, W., Beard, R. W., and Kingston, D. B., "Multi-Agent Kalman Consensus with Relative Uncertainty," *Proceedings of American Control Conference (ACC)*, Vol. 3, IEEE, Piscataway, NJ, June 2005, pp. 1865–1870.

[9] 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, No. 9, 2004, pp. 1520–1533.
doi:10.1109/TAC.2004.834113

[10] Alighanbari, M., and How, J. P., "An Unbiased Kalman Consensus Algorithm," *Proceedings of American Control Conference (ACC)*, IEEE, Piscataway, NJ, June 2006, pp. 3519–3524.

[11] Moallemi, C. C., and Roy, B. V., "Consensus Propagation," *IEEE Transactions on Information Theory*, Vol. 52, No. 11, 2006, pp. 4753–4766.
doi:10.1109/TIT.2006.883539

[12] Olshevsky, A., and Tsitsiklis, J. N., "Convergence Speed in Distributed Consensus and Averaging," *IEEE Conference on Decision and Control (CDC)*, IEEE, Piscataway, NJ, 2006, pp. 3387–3392.

[13] Ren, W., Beard, R. W., and Atkins, E. M., "Information Consensus in Multivehicle Cooperative Control," *IEEE Control Systems Magazine*, Vol. 27, No. 2, April 2007, pp. 71–82.
doi:10.1109/MCS.2007.338264

[14] Hatano, Y., and Mesbahi, M., "Agreement over Random Networks," *IEEE Transactions on Automatic Control*, Vol. 50, No. 11, 2005, pp. 1867–1872.

[15] Wu, C. W., "Synchronization and Convergence of Linear Dynamics in Random Directed Networks," *IEEE Transactions on Automatic Control*, Vol. 51, No. 7, 2006, pp. 1207–1210.
doi:10.1109/TAC.2006.878783

[16] Tahbaz-Salehi, A., and Jadbabaie, A., "A Necessary and Sufficient Condition for Consensus over Random Networks," *IEEE Transactions on Automatic Control*, Vol. 53, No. 3, 2008, pp. 791–795.

[17] Sariel, S., and Balch, T., "Real Time Auction Based Allocation of Tasks for Multi-Robot Exploration Problem in Dynamic Environments," *Proceedings of the AAAI-05 Workshop on Integrating Planning into Scheduling*, AAAI, Palo Alto, CA, 2005.

[18] 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," *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, IEEE, Piscataway, NJ, 2005.
doi:10.1109/KIMAS.2005.1427101

[19] Atkinson, M. L., "Results Analysis of Using Free Market Auctions to Distribute Control of UAVs," *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, AIAA Paper 2004-6558, 2004.

[20] Lemaire, T., Alami, R., and Lacroix, S., "A Distributed Task Allocation Scheme in Multi-UAV Context," *IEEE International Conference on Robotics and Automation (ICRA)*, Vol. 4, IEEE, Piscataway, NJ, 2004, pp. 3622–3627.

[21] Walsh, W., and Wellman, M., "A Market Protocol for Decentralized Task Allocation," *Proceedings of International Conference on Multi Agent Systems*, IEEE, Piscataway, NJ, 1998.
doi:10.1109/ICMAS.1998.699077

[22] Lagoudakis, M. G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., and Jain, S., "Auction-Based Multi-Robot Routing," *Proceedings of Robotics: Science and Systems*, MIT Press, Cambridge, MA, 2005, pp. 343–350.

[23] Amstutz, P., Correll, N., and Martinoli, A., "Distributed Boundary Coverage with a Team of Networked Miniature Robots Using a Robust Market-Based Algorithm," *Annals of Mathematics and Artificial Intelligence*, Vol. 52, Nos. 2–4, 2008, pp. 307–333.
doi:10.1007/s10472-009-9127-8

[24] Hoeing, M., Dasgupta, P., Petrov, P., and O'Hara, S., "Auction-Based Multi-Robot Task Allocation in COMSTAR," *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, ACM, New York, 2007, pp. 1435–1442.

[25] Sujit, P. B., and Beard, R., "Distributed Sequential Auctions for Multiple UAV Task Allocation," *Proceedings of American Control Conference (ACC)*, IEEE, Piscataway, NJ, 2007, pp. 3955–3960.

[26] Choi, H.-L., Brunet, L., and How, J. P., "Consensus-Based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, Vol. 25, No. 4, Aug. 2009, pp. 912–926.
doi:10.1109/TRO.2009.2022423

[27] Marden, J., and Wierman, A., "Overcoming the Limitations of Utility Design for Multiagent Systems," *IEEE Transactions on Automatic Control*, Vol. 58, No. 6, June 2013, pp. 1402–1415.
doi:10.1109/TAC.2013.2237831

[28] Garcia, E., and Casbeer, D. W., "Cooperative Task Allocation for Unmanned Vehicles with Communication Delays and Conflict Resolution," *Journal of Aerospace Information Systems*, Vol. 13, No. 2, 2016, pp. 1–13.
doi:10.2514/1.I010218

[29] Johnson, L. B., Choi, H.-L., Ponda, S. S., and How, J. P., "Allowing Non-Submodular Score Functions in Distributed Task Allocation," *Proceedings of IEEE Conference on Decision and Control (CDC)*, IEEE, Piscataway, NJ, Dec. 2012, pp. 4702–4708.

[30] Bertsimas, D., and Weismantel, R., *Optimization over Integers*, Dynamic Ideas, Belmont, MA, 2005, pp. 541–548.

[31] Fugishige, S., *Submodular Functions and Optimization*, 2nd ed., Annals of Discrete Mathematics, Elsevier Science, Amsterdam, The Netherlands, 2005, pp. 45–65.

[32] Krause, A., Guestrin, C., Gupta, A., and Kleinberg, J., "Near-Optimal Sensor Placements: Maximizing Information While Minimizing Communication Cost," *Proceedings of 5th International Conference on Information Processing in Sensor Networks*, ACM, New York, 2006, pp. 2–10.

[33] Ponda, S. S., Redding, J., Choi, H.-L., How, J. P., Vavrina, M. A., and Vian, J., "Decentralized Planning for Complex Missions with Dynamic Communication Constraints," *American Control Conference (ACC)*, Baltimore, MD, July 2010.

[34] Whitten, A. K., Choi, H.-L., Johnson, L., and How, J. P., "Decentralized Task Allocation with Coupled Constraints in Complex Missions," *Proceedings of American Control Conference (ACC)*, IEEE, Piscataway, NJ, June 2011, pp. 1642–1649.

[35] Lehmann, B., Lehmann, D., and Nisan, N., "Combinatorial Auctions with Decreasing Marginal Utilities," *Games and Economic Behavior*, Vol. 55, No. 2, May 2006, pp. 270–296.

[36] Shima, T., Rasmussen, S. J., and Chandler, P., "UAV Team Decision and Control Using Efficient Collaborative Estimation," *Proceedings of American Control Conference (ACC)*, IEEE, Piscataway, NJ, June 2005, pp. 4107–4112.

[37] Johnson, L., Choi, H.-L., and How, J. P., "The Hybrid Information and Plan Consensus Algorithm with Imperfect Situational Awareness," *Distributed Autonomous Robotic Systems: The 12th International Symposium on Distributed Autonomous Robotic Systems*, Vol. 112, Springer Tracts in Advanced Robotics, Jan. 2016, pp. 221–233.

[38] Johnson, L., Choi, H. L., and How, J. P., "The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial," *IEEE Control Systems Magazine*, Vol. 36, Aug. 2016, pp. 39–44.

[39] Johnson, L., "Decentralized Task Allocation in Communication Contested Environments," Ph.D. Dissertation, Massachusetts Inst. of Technology, Dept. of Aeronautics and Astronautics, Cambridge MA, Feb. 2016.

E. Atkins
Associate Editor