AFRL-AFOSR-VA-TR-2015-0321

Distributed Hybrid Information and Plan Consensus HIPC for Semi-autonomous UAV Teams

Jonathan How
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

09/18/2015
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTA2
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 09-14-2015	**2. REPORT TYPE** Final Performance Report	**3. DATES COVERED** *(From - To)* June 2014 - June 2015

4. TITLE AND SUBTITLE Distributed Hybrid Information and Plan Consensu HIPC for Semi-autonomous UAV Teams	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER FA9550-11-1-0134
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Jonathan How	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Massachusetts Institute of Technology 77 Massachusetts Avenue Cambridge, MA2139	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research; Boston Regional Office/N62879 495 Summer Street, Room 627 Boston, MA 02210-2109	**10. SPONSOR/MONITOR'S ACRONYM(S)** ONR
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) N/A

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The research objective of this project was to increase the capabilities of decentralized task allocation algorithms. Progress has been made on several different fronts, including: a)~chance-constrained task allocation, b) task allocation with tasks defined as Markov Decision Processes, c) guaranteeing network connectivity during mission execution, d) allowing the use of non-submodular score functions during the decentralized allocation, and e) decreasing the convergence time by utilizing all of the information available in the network, f) hardware results that demonstrate the difficulty of planning in communication contested environments and the utility of using LICA algorithms, and g) a tutorial on the basics of decentralized task allocation for a general audience is currently in revision for Control Systems Magazine. Combining these results has significantly improved the state of the art capabilities of decentralized task allocation and work continues to refine these approaches.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**		58	Jonathan How
					19b. TELEPHONE NUMBER *(Include area code)* 617-253-3267

Reset

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18
Adobe Professional 7.0

Distributed Hybrid Information and Plan Consensus for Semi-autonomous UAV Teams

Final Report

AFOSR # FA9550-11-1-0134

Jonathan P. How
Laboratory for Information and Decision Systems
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology

Abstract

The research objective of this project was to increase the capabilities of decentralized task allocation algorithms. Progress has been made on several different fronts, including: a) chance-constrained task allocation [1], b) task allocation with tasks defined as Markov Decision Processes [2], c) guaranteeing network connectivity during mission execution [3], d) allowing the use of non-submodular score functions during the decentralized allocation [4, 5], and e) decreasing the convergence time by utilizing all of the information available in the network [6–8], f) hardware results that demonstrate the difficulty of planning in communication contested environments and the utility of the proposed approach presented in [6–8], and g) a tutorial on the basics of decentralized task allocation for a general audience is currently in revision for Control Systems Magazine [9]. Combining these results has significantly improved the state of the art capabilities of decentralized task allocation and work continues to refine these approaches.

1 Research Results

The following summarizes the main research results from the project:

1. The first contribution proposed new risk allocation strategies for distributed chance-constrained planning in multi-agent multi-task stochastic missions [1, 10]. Building on previous efforts that extended chance-constrained planning to distributed environments [10, 11], this work presented a more formal approach to risk allocation, and proposed several risk allocation strategies for homogeneous and heterogeneous agents that can be leveraged within the distributed chance-constrained planning framework. In particular, the contributions of this work included: presenting a framework for homogeneous and heterogeneous risk allocation, proposing risk allocation strategies that exploit domain knowledge of agent score distributions to improve team performance, and providing insights and intuition as to what parameters affect these allocations and what features affect the performance of the overall chance-constrained mission score given the distributed approximation. Results demonstrated improved performance in time-critical mission scenarios given allowable risk thresholds.

2. This work developed a multiagent task allocation algorithm for general stochastic tasks based on Markov Decision Processes [2]. This provides a task allocation algorithm that is scalable to situations involving a large number of agents and tasks, while capable of incorporating the generality of MDP models of those tasks. The main intuition behind

the formulation of this approach is that the value function of an MDP task evaluated at the expected starting conditions for an agent can be used as the score function for task allocation. Results of this work showed the feasibility of using these methods in realistic mission scenarios including environments whose tabular representation would have more than 10^{100} states.

3. Research also investigated how to construct cooperative distributed planning algorithms that ensure network connectivity for a team of heterogeneous agents operating in a dynamic communication limited environment [3, 12, 13]. By employing underutilized agents as communication relays, the CBBA with Relays algorithm improves network connectivity and team range, without limiting the scope of the active agents, thus improving mission performance. The algorithm builds upon a distributed task allocation framework, named the Consensus-Based Bundle Algorithm (CBBA), and leverages information available through existing consensus phases of CBBA to predict the network topology. If constraint violations are detected, relay tasks to repair connectivity are created and integrated into the algorithm in a consistent manner. A key feature of CBBA with Relays is that the network topology is computed only at select mission critical times, and local information already available to each agent is leveraged in this prediction. As a result, the algorithm is able to preserve the distributed and polynomial-time guarantees of CBBA, making it well suited to real-time applications. Algorithm performance is validated through simulation trials and through experimental indoor and outdoor field tests.

4. Additionally, research expanded the class of score functions that can be used with decentralized task allocation algorithms. Traditional methods have relied on submodularity, a powerful property that can be exploited for provable performance and convergence guarantees in decentralized task allocation algorithms. However, some mission scenarios cannot easily be approximated as submodular a priori. This work introduced an algorithmic extension for distributed multi-agent multi-task assignment algorithms that enables them to converge using non-submodular score functions [4] and is extended in a submission to IJRR [5]. These enhancements utilized non-submodular ranking of tasks within each agent's internal decision making, while externally enforcing that shared bids *appear* as if they were created using submodular score functions. Convergence and performance bounds are proved for this algorithm called Bid-Warped CBBA (BW-CBBA). The results of this effort showed significant improvements over the previously available approaches of hand-tuned heuristics that approximate the true non-submodular score functions.

5. Recent work has developed the Hybrid Information and Plan Consensus Algorithm (HIPC) [6–8]. This algorithm uses implicit coordination to plan for a subset of the team on-board each agent, then uses plan consensus to fix conflicts in the assignment constraints that may arise. By combining the ideas of local plan consensus and implicit coordination the algorithm empirically reduced the convergence time and number of messages required for distributed task allocation problems. Recent work [7] rigorously proves convergence and provides a worst case convergence bound that is no slower than bid warped CBBA [4], requiring 2 times the number of assigned tasks times the network diameter ($2N_t\mathcal{D}$) iterations. Further work expanded this to imperfect situa-

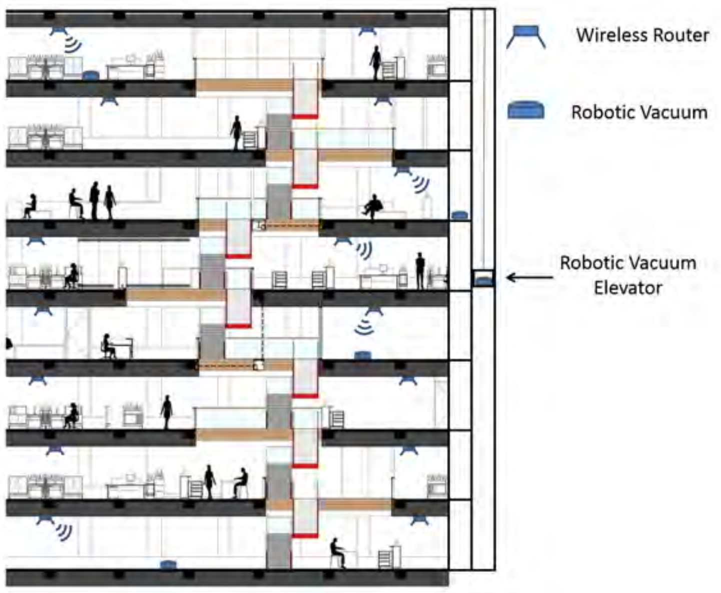


Figure 1: (Example 1) Cleaning a large building using robotic vacuum cleaners. The building contains a robust communication network so centralized planning approaches are appropriate.

tional awareness [8] and a real time hardware demonstration was conducted showing the validity of this approach.

6. A submission in review for the Control Systems Magazine [9] provides an introduction to decentralized task allocation. It specifically introduces the material in tutorial form, with specific examples identifying the components of particular missions that force the use of decentralized algorithms. The goal of this work is introduce our view of decentralized task allocation to the broadest audience possible and to provide a vocabulary for discussing different decentralized approaches.

The following sections provide more detail on some of the newer work that has been developed, specifically from [5, 8, 9].

2 General Planning Examples

This section will introduce some canonical examples of planning problems to give some intuition about the main concepts that are considered in this work.

2.1 Example 1: Office cleaning

High Level Mission: clean the floors of a large office building using autonomous robotic vacuums.
Agents: robotic vacuum cleaners and a robotic elevator that can move the vacuum cleaners between the floors of the building. The vacuum robots will have on-board sensors with the capabilities of mapping the floor to be cleaned, and measuring the "cleanliness" of the floor to make sure the environment is clean before moving on.
Environment: 8 story building with full wireless network coverage. The mission environ-

ment is also assumed to be dynamic because 1) the furniture may move around a bit on a daily basis, 2) workers in different parts of the building may stay later on some nights forcing a delay in entering these areas, and 3) some areas may need heavier cleaning some days.
Detailed Objectives: optimize the allocation and motion of each of the vacuum cleaners to efficiently clean the space while both minimizing resource consumption (i.e. robot power, elevator usage) and minimizing inconvenience to humans working in the building.
Planner considerations: The computational problem is to decide what agents clean what rooms and when, and how the agents move while they are cleaning those rooms. The simplest planner available in terms of real time complexity would be to construct a single static *plan* that is executed every building cleaning cycle. Given that the environment is assumed to be dynamic, this approach will not work well in practice because the environment will likely never match the one assumed by the plan. Another off-line approach could be to construct a *policy*, which provides a detailed plan for the agents to execute for every possible realizable environment. This approach can work well if all possible environments can be modelled efficiently a priori. For most non-trivial environments, however, these off-line policies become intractable to compute (or even store) because there are too many possible scenarios that can be encountered. When this is the case, it is necessary to consider at least partial on-line approaches.

When constructing on-line planning approaches, computational time becomes very important (because the time you spend computing is time not servicing the overall mission.) One of the most common simplifying assumptions is decoupling the assignment of the agents (what rooms do they clean and when), from the motion plans of these agents when they are in the rooms. This is a good assumption when the relevant information about the motion plan can be predicted without having to compute agent trajectories. In this context, if the problem is partitioned into cleaning rooms, the only information the allocation planner would need, is the location of the agent before and after it has cleaned the room, and how long it will take to clean the room. As long as the optimal plan for cleaning the building w.r.t. the detailed objective function, does not have individual agents cleaning the same room simultaneously, this simplification does not sacrifice performance of the overall mission. For this environment, decoupling the agent motion inside the rooms from the room assignments seems reasonable because the vacuums will likely enter, and exit rooms through doorways, and have estimates about how long it will take to clean rooms.

Solving the motion planning part of this problem is not the focus of this tutorial, as such, it is assumed that the agents have the ability to plan their trajectories through the rooms, and accomplish the cleaning tasks for those rooms.

The process by which the allocation assignments are constructed will be the primary focus of this paper. Given that the building wireless network provides a relatively robust communication network, a **centralized task allocator** that is aware of the states and capabilities of all agents can be run in real time. In comparison to the next example that will be presented, there is no need to decentralize the assignment planning aspects of this problem because the communication network is so good, and the vacuum cleaners don't need to change their behavior so often that a centralized planner would struggle to provide assignments in a reasonable time.

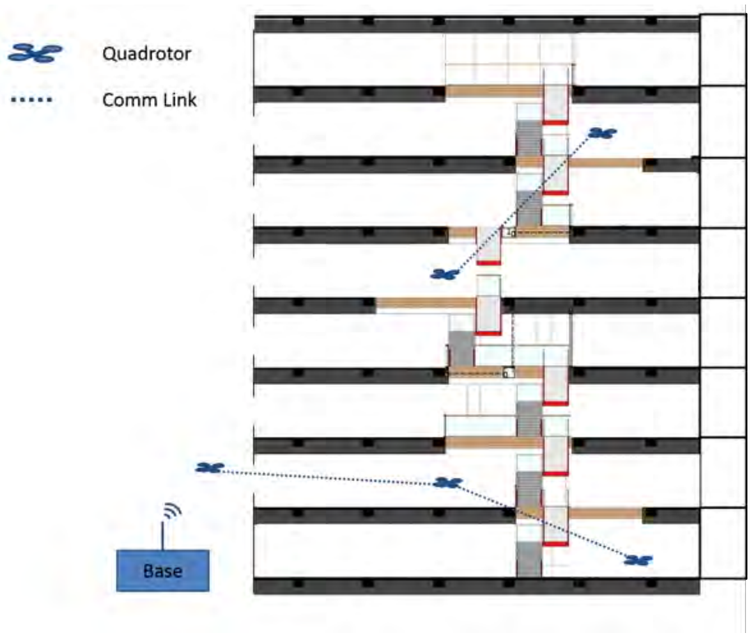


Figure 2: (Example 2) Surveying a large building using quadrotor helicopters without a building communication infrastructure. This environment will likely need some form of decentralized planning to ensure mission completion with the required ad hoc network.

2.2 Example 2: Emergency building survey

High Level Mission: search a large building in an emergency situation. This involves mapping the current physical state of the building, and potentially identify key features that require actions from a base station (i.e finding victims that may be trapped, structural problems, etc.)
Agents: flying reconnaissance robots (shown as quadrotor helicopters.) The vehicles use on-board sensors to do the required mapping as well as provide feedback for their own motion control. The agents use radios to communicate with each other and to the base station.
Environment: it is assumed that a good floor plan of the building is unavailable or the emergency situation has potentially caused structural damage to the building and the effective floor plan has changed. Additionally it is assumed that there is not a reliable building wifi network or an elevator system to move ground robots around, requiring the use of flying robots and communication over an ad hoc network between the agents. An additional complication is that radio communications between the agents may be obstructed so communication may fail in certain vehicle geometries.
Detailed Objectives: search building as fast as possible while possibly ensuring important information gets relayed to the base station quickly.
Planner considerations: Using similar logic to Example 1, it will be virtually impossible to use an offline plan or policy approach and expect good performance. There is very little known about the environment before the mission is being executed, so real time adjustments based on the observed environment will be essential. There are, however, significant differences in the two examples. First, this domain is much more time sensitive, meaning planning decisions must be made very quickly. Additionally, the communication network is much less

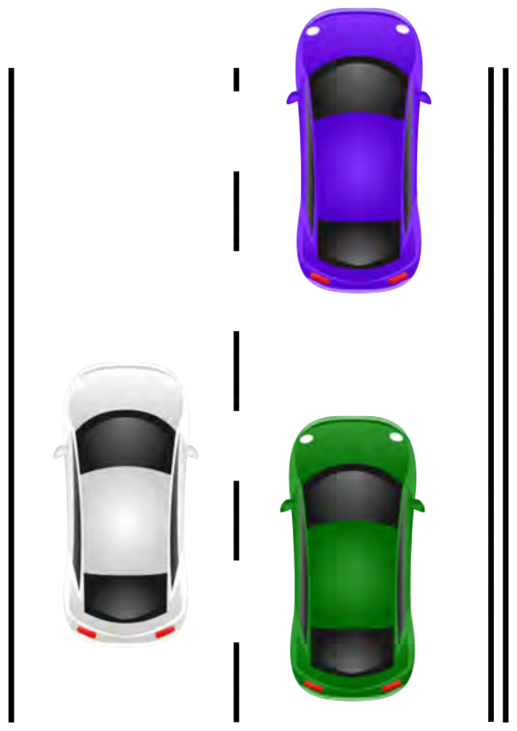


Figure 3: (Example 3) An autonomously controlled car on the left is attempting to merge into a congested right lane with two uncontrolled vehicles. This problem does not fit into the typical multi agent planning framework that is discussed in this tutorial.

favorable so connections to a centralized location are not even guaranteed. Therefore, the agents will have to utilize an ad hoc network between themselves to coordinate their actions and plan their trajectories throughout the building. Complicating the communication environment further, the ad hoc network will also be dynamic during the mission execution as agents move around in the building. Since communications to the base are not guaranteed and decisions must be made on a relatively quick time scale it doesn't make sense for the agents to re-establish global communication before making decisions. This scenario will likely require decentralized algorithms to sufficiently satisfy the mission objectives. There are many different algorithmic forms that could be utilized for this planning assignment problem, some of which will be discussed throughout this tutorial.

The majority of the paper will focus on environments that look like some combination of the first two examples. The purpose of separate examples is to highlight the transition in the communication and planning environment needed to motivate utilizing a decentralized planning approach. The purpose of Example 3, as will be shown next, is to highlight that there are some problems that are not modeled well in typical planning frameworks.

2.3 Example 3: Autonomous Lane Merging

High Level Mission: merging lanes in an autonomous vehicle.
Agents: a single autonomous vehicle.
Environment: normal highway driving. In this scenario the cars in the right lane are considered as part of the environment because we don't have direct control over them. There

can only be social cues (i.e. turning on a turn signal) do indicate a desire to merge into that lane.

Detailed Objectives: The controlled agent in the left lane wants to merge into the right lane as quickly as possible while remaining safe. An important note here is that the cars in the right lane have no direct incentive to let or agent merge.

Planner considerations: The key issue with this problem is that there are two autonomous vehicles that have no explicit incentive to coordinate with the merging car. The only reason a car in the right line would slow down is due to some social behavior. Understanding this behavior is handled in the field of game theory and is based on making decisions based on reciprocal behavior. The only real option for an agent attempting to use a classical planning algorithm would be to model the other agents as part of the environment and try to manipulate the environment, with signals or changing speeds to signal a desire to merge. This would involve predictions of the environment dynamics and how these agents would respond to this behavior which is just another way of looking at game theoretic decision making. This is a severe increase in complexity that will not be addressed in this tutorial. The take away from this example is thatthere exists problems that are not centralized, but do not permit the use of the typical decentralized approaches that will be presented here.

3 Decentralized Task Allocation

An important contribution of this work is utilizing decentralization as a tool that can improve mission performance. This contrasts the way it is typically viewed as simply a feature of a particular algorithm. Just because something *can* be decentralized, doesn't mean that it should and certainly doesn't mean that it will improve performance. In much of the literature the reasons for decentralizing algorithms are lost and one of the primary goals of this work is to provide intuition on when decentralization should be used, and when and what it actually helps to decentralize [14].

To start this discussion, it is important to first recap what type of problems are solved best in centralized environments. An *ideal* centralized environment requires 1) agent objective knowledge is available at a centralized location, 2) communication channels from the centralized location to each agent, and 3) full information state knowledge at the centralized location. Either the information environment or the communication environment can cause performance degradation in a purely centralized task allocation algorithm. For *ideal* decentralized behavior, two conditions must be met: 1) Agent decisions cannot conflict with other agents, and 2) the relevant agent information state must be locally observable. All other environments introduce a degradation from the ideal performance of these two paradigms.

After understanding what types of problems centralized and decentralized algorithms solve well, we can start to discuss how to utilize decentralization (or equivalently centralization) to improve performance. Figure 4 highlights how different paradigms can achieve different results for a fixed bandwidth. The design criteria for creating the best algorithms for a given environment becomes, what information is communicated to whom, where is the computation done, and what is computed. The goal is then to design algorithms that can answer these questions during runtime for the realized mission and environment. The answers to these questions have led to the ideas in the Hybrid Information and Plan Consensus algorithm (HIPC).

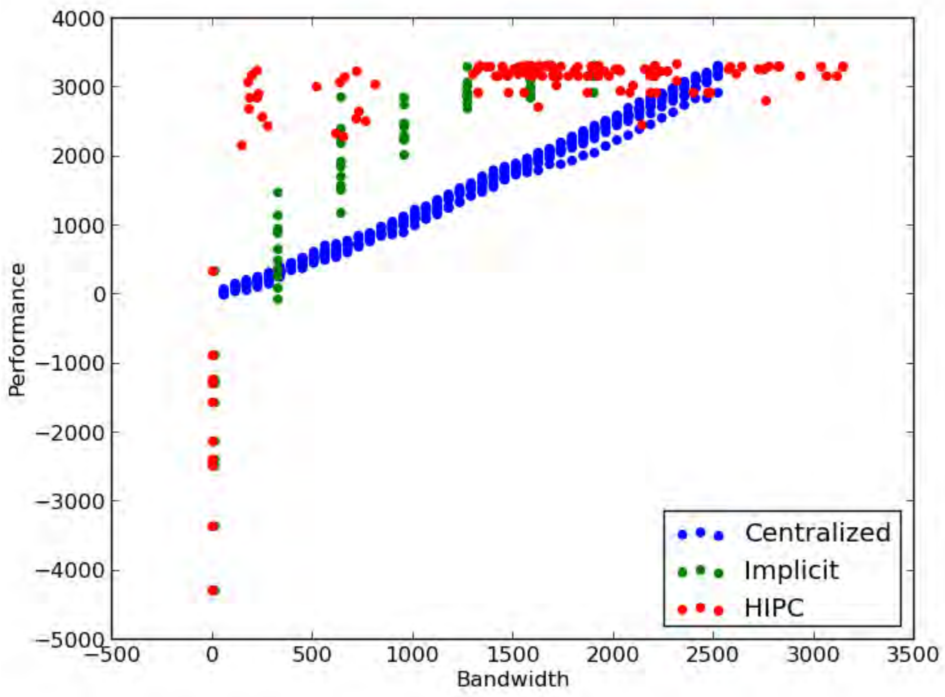


Figure 4: Comparison between three different planning approaches over a wide range of bandwidth restrictions and problem scenarios. The blue dots show the progression of planner performance for centralized planners as bandwidth increases, the green dots show the planner performance of implicit coordination, and the red dots show the performance that HIPC achieves. This figure is essentially a cartoon that illustrates an environment where utilizing hybrid approaches, can provide good solutions much faster in some communication environments.

3.1 Understanding How Information is Shared between Agents

The goal of standard multi-agent task allocation algorithms [15–17] is to coordinate a team of cooperative agents 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 (i.e. fuel, power, vehicle capabilities, etc.) Centralized task allocation algorithms are typically preferred when an application requires high levels of collaboration. However, in contested environments where communications may be unavailable, unreliable, have high latency, or high cost, relying on centralized solutions can be impractical. In these communication limited environments, it is necessary to consider distributed or decentralized algorithms [18]. Unfortunately, using distributed or decentralized algorithms usually introduces additional complications, including difficulties establishing both algorithmic convergence and performance. The major reason for these complications is that in decentralized environments, agents may operate on only partial information, and 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 categorized into two main information assumptions: global information consistency and local information consistency.

1. *Global information consistency assumptions* (GICA) 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 team-wide truth. Given that these global information consistency assumptions require recognizing information across the entire team, reaching information consistency happens on a global communication time scale. Algorithms that utilize these assumptions can be found in Refs. [17, 19–37].

2. *Local information consistency assumptions* (LICA) do not require global consistency of any information. Only requiring local information consistency can provide a much shorter time-scale for utilizing new information (than 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 is that agents cannot guarantee any piece of information is globally consistent and thus algorithms utilizing these assumptions must take this into account during the planning process [38–41].

There are many factors that determine whether requiring global or allowing only local information consistency will provide the best 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 the added complexity of local information consistency assumptions. Conversely, a trivial example of when local information consistency assumptions are necessary is when the network can become temporarily disconnected. Network disconnections during global information consistency algorithms will break the assumptions tied to performance and sometimes convergence. Possible repairs would include waiting until the network reconnects (may never happen) or detecting the new network, continuing on and *hoping* that future network dynamics wont break the resulting allocation (which is not a guarantee). This domain is handled naturally with local consistency assumptions which can ensure provably good performance in contested communication environments.

In other domains where global consistency assumptions are reachable, local consistency algorithms can actually drastically reduce convergence times. Figure 5 shows that the convergence time of sequential auction algorithms (such as one described in [42]) will require roughly the number of tasks times the network diameter iterations to reach convergence. Even algorithms that globally consider bundles of tasks simultaneously (e.g. a GICA version of CBBA [40] 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 [40]). 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.

These insights imply that some environments are well suited for algorithms that only use only local information consistency assumptions (referred to as LICA for the rest of the report). Therefore, this work investigates the objective function limitations that LICA algorithms impose on assignment convergence and performance. In general, issues arise because

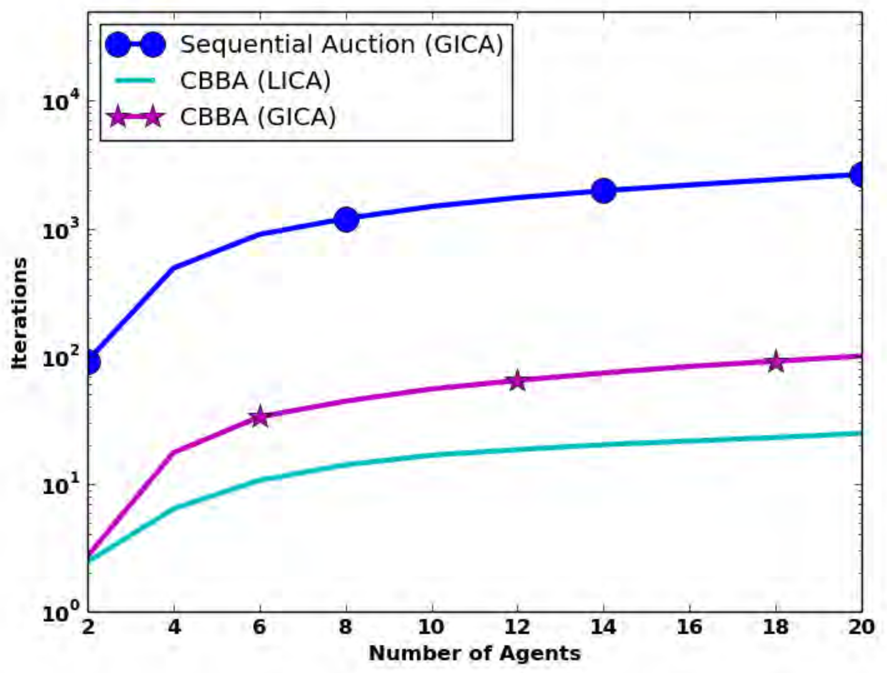


Figure 5: Compares the number of iterations required for convergence in 500 Monte Carlo trials of a 200 task mission with a varying number of agents.

algorithms that make decisions based only on LICA cannot trust that their information is globally accurate. This means there is no mechanism to 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 [38–41]. Unfortunately, many objective functions of interest, including those that incorporate fuel penalties, information gathering metrics, cooperative tasking metrics, and stochastic objectives may take a non-submodular form. A solution to this is outlined in this work if functions are *nearly* submodular. The approach is to modify the information shared between agents so that it looks as if each agent was using a submodular objective function, even though agent preferences will be determined using non-submodular objectives. This approach will provide convergence guarantees for all objective functions, and will quantify when non-trivial performance bounds exist.

The first algorithmic contribution of this work was the creation of BW-CBBA [4], which is a local information consistency algorithm that converges with all forms of objective functions. The approach taken in this algorithm is to modify the information shared between agents so that looks as if it were created using a submodular objective function, even though agent preferences are still determined by a non-submodular objective function. This approach will provide convergence guarantees for all score functions, and will quantify when non-trivial performance bounds are possible with non-submodular objective functions.

3.2 Problem Formulation

This section presents the general problem statement and formalizes the class of problems being solved by HIPC. Given a set of N_a agents and N_t tasks, the goal of the task allocation algorithm is to find an assignment of tasks to agents that maximizes the global reward. The global objective function for the mission is given by a sum over local objective functions for each agent, while 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:

$$\begin{aligned} \max_{\mathbf{x},\boldsymbol{\tau}} \quad & \sum_{i=1}^{N_a}\sum_{j=1}^{N_t} F_{ij}(\mathbf{x},\boldsymbol{\tau})\ x_{ij} \qquad (1)\\ \text{s.t.} \quad & \mathbf{H}(\mathbf{x},\boldsymbol{\tau}) \leq \mathbf{d} \\ & \mathbf{x} \in \{0,1\}^{N_a\times N_t},\ \boldsymbol{\tau} \in \{\mathbb{R}^+ \cup \emptyset\}^{N_a\times N_t} \end{aligned}$$

where $\mathbf{x} \in \{0,1\}^{N_a\times N_t}$, 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; $\boldsymbol{\tau} \in \{\mathbb{R}^+ \cup \emptyset\}^{N_a\times N_t}$ is the set of real-positive decision variables τ_{ij} indicating when agent i will service its assigned task j (where $\tau_{ij} = \emptyset$ 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 $\mathbf{H} = \left[\mathbf{h}_1 \dots \mathbf{h}_{N_c}\right]^T$, with $\mathbf{d} = \left[d_1 \dots d_{N_c}\right]^T$, define a set of N_c possibly nonlinear constraints of the form $\mathbf{h}_k(\mathbf{x},\boldsymbol{\tau}) \leq d_k$ that capture transition dynamics, resource limitations, etc. This problem formulation can accommodate most design objectives and constraints used in multi-agent decision making problems (e.g. search and surveillance missions where F_{ij} represents the value of acquired information and the constraints $\mathbf{h}_k$ capture fuel limitations and/or no-fly zones, or rescue operations where F_{ij} is time-critical favoring earlier τ_{ij} execution times, etc). An important observation is that, in Equation (1), the scoring and constraint functions explicitly depend on the decision variables $\mathbf{x}$ and $\boldsymbol{\tau}$, which makes this general mixed-integer programming problem (NP-hard) [43].

The algorithms used in this work solve *distributed greedy multi-agent multi-assignment* problems. For each agent, these problems take a similar form to Equation (1), except individual agents independently create assignments for themselves, and then iterate with others using a consensus algorithm to produce a final team-wide assignment. The details of this process will be explored throughout the rest of this report.

3.3 Submodularity

The shape of the mission objective function is fundamentally related to the difficulty of creating good task assignments. Specifically, objective functions must be submodular [44] for LICA algorithms to converge and have non-trivial performance bounds [38–41]. For the purposes of this paper, submodularity can be defined as follows: the marginal score function $F(S|\mathcal{A})$ for incrementing an allocation by a set S to an existing task allocation $\mathcal{A}$, must satisfy the following

$$\begin{gathered} F(S|\mathcal{A}') \geq F(S|\mathcal{A}) \qquad (2)\\ \forall A'\ s.t.\ \mathcal{A}' \subset \mathcal{A} \end{gathered}$$

To guarantee that F is a set function in this notation, scores are defined as incremental assignment S's marginal contribution to the task allocation environment $F(S|\mathcal{A}) = F(S \cup \mathcal{A}) - F(\mathcal{A})$. A task environment $\mathcal{A}$ can be thought of as a globally consistent set of allocations S, where each element $s \in S$ defines an assignment of tasks to agents. 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 [45]), many do not. As will be shown in the next section, submodular score functions are essential for LICA algorithms to guarantee convergence. Previous work identified modifications to the score functions that employ heuristics to ensure that submodularity is satisfied [40], 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 (CBBA) 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.

4 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 2 tasks and 2 agents. Consider the following algorithm: each agent sequentially produces bids on a set of available tasks, then 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 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 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).

Example 1: Baseline Submodular Score Function

This example 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 4 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 δ less than the bid for the first task. The bid values in this example could have been produced by a submodular score function since 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.

Example 1: Allocations with a submodular score function *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

`Iteration 1`

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

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

`Iteration 2`

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

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

Example 2: Non-submodular score function

Example 2 highlights how convergence is lost when non-submodular score functions are introduced with the algorithm defined at the beginning of Sec. 4. In this example, δ may take any value, $0 < \delta < \infty$. At `Iteration 1` both agents choose between 4 feasible bundles, and they choose the bundle with the highest total score (Agent 1 chooses bundle 4 and Agent 2 chooses 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 Equation (2).

Between `Iteration 1` and `Iteration 2` the agents share their bids with each other. Agent 1 is outbid on task 1, and 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, and 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 and `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-

Example 2: Allocations with a non-submodular score function
Bundle Creation

Agent 1 candidate bundles

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

Agent 2 candidate bundles

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

Algorithmic progression

Iteration 1

Agent 1: $\{(1,T),(2,T+\delta)\}$
Agent 2: $\{(2,T),(1,T+\delta)\}$

Iteration 2

Agent 1: $\{\}$
Agent 2: $\{\}$

Iteration 3

Agent 1: $\{(1,T),(2,T+\delta)\}$
Agent 2: $\{(2,T),(1,T+\delta)\}$

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 an agent has entered into one of these cycles, 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.

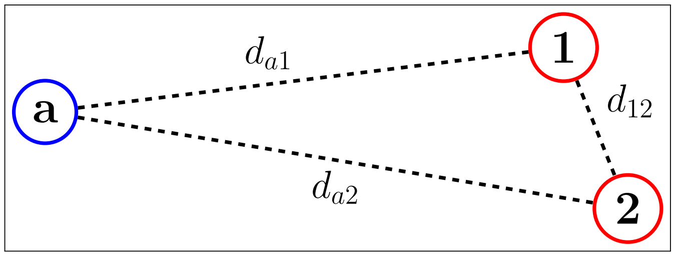


Figure 6: A potential task environment described in Example 3, where **a** represents an agent, and **1** and **2** are task locations. The notation d_{uv} is the distance required to move from location u to location v.

Example 3: Waypoint tasks

Consider the potential task scenario illustrated in Figure 6 involving one agent (blue circle labeled **a**) and two tasks (red 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 $F_{ij}(\mathbf{x}, \boldsymbol{\tau})$ for this environment is defined as follows:

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

where $F_{ij}(\mathbf{x}, \boldsymbol{\tau})$ is the score for assigning task j to agent i given current assignment $\mathbf{X}$ and arrival timings $\boldsymbol{\tau}$, R is the reward obtained for servicing a task, f_i is the fuel penalty per unit distance for agent i, and $d_{\mathbf{x} \oplus j}$ is the increase in distance travelled by inserting task j into the current assignment $\mathbf{x}$ (which by assumption cannot already include an assignment on task j by agent i). If the LICA algorithm introduced at the beginning of Sec. 4 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 Equation (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 [46]. For example, the score function (3) can be approximated as:

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

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. However, this score function cannot capture the fact that, since 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

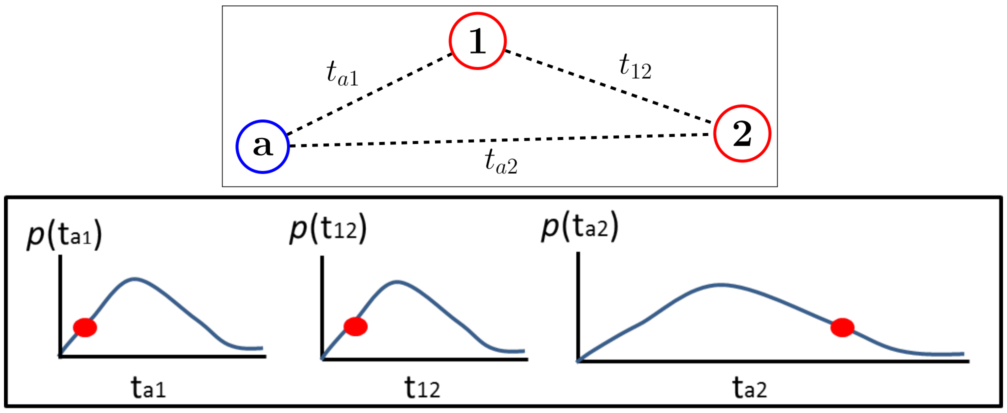


Figure 7: A potential task environment described in Example 4, where **a** represents the agent, and **1** and **2** are task locations. The notation t_{uv} is travel time measured from location u to location v. The three graphs show the probability distributions of the travel times for each of the three legs in the example (t_{a1},t_{12}, and t_{a2}.) The red dots on these distributions show a candidate sampling locations where $t_{a1} + t_{12} < t_{a2}$.

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 Section 6.3 and these demonstrate the potential downside of using a priori submodularized functions.

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 7 illustrates an example where non-submodular effects can arise due to this necessary sampling.

Assume that the score function has the simple form:

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

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. 7) $\mathbf{t}_{a2}$ can be greater than the sum of $\mathbf{t}_{a1}$ and $\mathbf{t}_{12}$,

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

Therefore, even though the expected mission scores are submodular,

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

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

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

In this notation, $\mathbf{t}^{[m]}$ is the mth sample from the random variable $\mathbf{t}$. 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.

4.1 Convergence for non-submodular score functions

The convergence failures highlighted above 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 [40] 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 0 on either task 1 or 2. This is not a desired result; the approach provided in this paper prevents algorithmic cycling by pre-emptively changing the bid values, rather than relying on the performance-degrading process of identifying cycles after they happen.

4.2 Using Local Information Consistency Assumption Algorithms

The content of this section has 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. 2, there exist environments that require or could utilize LICA task allocation algorithms to improve mission performance. The rest of this paper outlines 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.

5 LICA Algorithms

This section focuses on a new description of the consensus based bundle algorithm [40] as well as algorithmic modifications specifically related to handling non-submodular score functions. The following description of CBBA is a slight modification of the one that was originally proposed in [40], introducing some new terms which will be used for proving relevant aspects of the new algorithm proposed in this paper.

5.1 Baseline CBBA

CBBA is a local information consistency assumption auction algorithm that provides provably good task assignments for multi-agent, multi-task allocation problems. The algorithmic structure of 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, $\mathbf{b}_i = (s_{ij_1}, \ldots, s_{ij_n})$ 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 $\mathcal{A} = \{s_{i_1j_1}, \ldots, s_{i_Nj_N}\}$, 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 non-identical 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.

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.

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 $\mathbf{b}_i \leftarrow \emptyset$
 6:     while $|\mathbf{b}_i| < L_t$ do
 7:         $\mathcal{J} \leftarrow \{j \mid s_{ij} \notin \mathcal{A}_i\}$
 8:         for all $j \in \mathcal{J}$ do
 9:             $c_{ij} \leftarrow F_{ij}(\mathbf{b}_i)$,
10:             $h_{ij} \leftarrow \prod_{s_{i'j} \in \mathcal{A}_i} \mathbb{I}(c_{ij} > c_{i'j})$
11:         end for
12:         $j^\star \leftarrow \operatorname{argmax}_{j \in \mathcal{J}} c_{ij} \cdot h_{ij}$
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:             $\mathbf{b}_i \leftarrow \mathbf{b}_i \oplus s_{ij^\star}$
16:             $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup s_{ij^\star}$
17:         else
18:             break
19:         end if
20:     end while
21:     return $(\mathbf{b}_i, \mathcal{A}_i)$
22: end procedure
```

The two phase algorithm then begins in the Bundle Building.

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 $\mathbf{b}_i$ and local bid space $\mathcal{A}_i$ (lines 3,5 of Alg. 1). This step is required for the performance guarantees of the algorithm[1], but in most cases the agent will re-add each of the tasks it has just dropped.

2. A local internal score function $F_{ij}(\mathbf{b}_i)$ is defined for each agent i and task j. It is a function of the agent's current bundle $\mathbf{b}_i$ and implicitly a function of the assignment constraints $\mathbf{G}(\mathbf{x}, \boldsymbol{\tau}) \leq \mathbf{d}$ posed in the problem formulation (Eq. 1). If a proposed assignment will not satisfy the constraints required by Eq. 1, $F_{ij}(\mathbf{b}_i)$ will return a value of $-\infty$. For complete notation consistency with Eq. 1 in Sec. 3.2, assume that there is a one-to-one mapping between $(\mathbf{b}_i)$ and $(\mathbf{x}, \boldsymbol{\tau})$. The complexity of actually

[1] 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.

picking the execution times τ_{ij} is not a focus of this paper, but other solutions used for CBBA can be implemented [47]. For each task j available in the environment (line 7), each agent i uses its local internal score function $F_{ij}(\mathbf{b}_i)$, 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 replaces the current 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 is reached, at which point the bundle building phase terminates. It is worth noting that in this formulation the values c_{ij} are used to rank the tasks, and the same values are used to construct the bids s_{ij}, which are shared with the other agents. The main result of this paper is to separate these two values in order to enable a larger class of score functions to converge with this type of algorithm.

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 time-stamp information, is then passed through a decision table (see [40], 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 re-enters the bundle building phase and the algorithm continues.
Score Function Fundamental to all of the convergence and performance guarantees for CBBA is that it must use a diminishing marginal gains (DMG) satisfying score function. The requirement of DMG for the CBBA score function is a special case of requiring submodularity as it was introduced in the Section 3.3 because it was defined for a specific marginal contribution to the existing bundle $\mathbf{b}_i$ as opposed to for all sets as was defined in Eq. 2. It was recognized in the seminal description of CBBA [40] but is updated here with the notation of bids and bundles. DMG is defined as

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

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

5.2 Bid Warping

The approach introduced in this work 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

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

$$\bar{c}_{ij} = \min\{c_{ij}, \min_{k \in \{1,\ldots,|\mathbf{b}_i|\}} c_{ij_k}\} \tag{11}$$

where c_{ij_k} is the unwarped score of the kth element in the current bundle, and $|\mathbf{b}_i|$ 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 $\bar{c}_{ij_{|\mathbf{b}_i|}}$

$$\bar{c}_{ij} = \min\{c_{ij}, \bar{c}_{ij_{|\mathbf{b}_i|}}\}. \tag{12}$$

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 $s_{ij} = \langle i, j, c_{ij} \rangle$:*

$$\begin{aligned} s_{i_1 j_1} \succ s_{i_2 j_2} \implies & c_{i_1 j_1} > c_{i_2 j_2} & (13)\\ & \vee\ c_{i_1 j_1} = c_{i_2 j_2}\ \&\ i_1 < i_2 & (14)\\ & \vee\ c_{i_1 j_1} = c_{i_2 j_2}\ \&\ i_1 = i_2\ \& \\ & \quad s_{i_1 j_1} \textit{ earlier in bundle than } s_{i_2 j_2} & (15) \end{aligned}$$

*These three conditions can completely define a strict ordering over bids. The first **or** clause (Eq. 13) defines that if the score $c_{i_1 j_1}$ of $s_{i_1 j_1}$ is larger than the score $c_{i_2 j_2}$ of $s_{i_2 j_2}$ then $s_{i_1 j_1} \succ s_{i_2 j_2}$. The second **or** clause (Eq. 14) defines that if the scores are the same $c_{i_1 j_1} = c_{i_2 j_2}$ 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 id's 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.*

5.3 Bid Warped CBBA

This section presents the main algorithmic modifications required for CBBA to use non-submodular score functions, the result of which is called Bid Warped CBBA (BW-CBBA).

Bid Warped CBBA: (Alg. 2: `BW-CBBA`) This algorithm 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 $\mathbf{b}_i^o$ and its local understanding of the global bid space $\mathcal{A}_i$. In order to initialize the algorithm, the convergence counter for agent i, denoted by k_i, is set to 0 (line 2 in Alg. 2); its **broadcast queue** Q_i, which is a list of pairs $\langle s, t \rangle$ consisting of a bid s and a time

Algorithm 2 BW-CBBA: Bid Warped CBBA

```
 1: procedure BW-CBBA($\mathbf{b}_i^o, \mathcal{A}_i$)
 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:         $\mathbf{b}_i, \mathcal{A}_i \leftarrow$ BW-BB($\mathcal{A}_i$)
12:         $\mathcal{A}_i, Q_i, \mathcal{Z}_i \leftarrow$ BW-TC($\mathbf{b}_i^o, \mathbf{b}_i, \mathcal{A}_i, Q_i, \mathcal{Z}_i$)
13:         $\mathbf{b}_i^o \leftarrow \mathbf{b}_i$
14:     end while
15:     return ($\mathcal{A}_i$)
16: end procedure
```

stamp t, is set to empty (line 3); and its **local time stamp matrix** $\mathcal{Z}_i(i', j)$, which records the time stamp of the most recent information about a bid made on task j by agent i', is initialized to all zeros (line 4).

2. While 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).

Bid Warped Bundle Building (Alg. 3: `BW-BB`) Again, for each agent i, the bundle building phase is run independently.

1. All current tasks in agent i's bundle $\mathbf{b}_i$ and tasks won by agent i in its local bid space $\mathcal{A}_i$ (lines 3,5 in Alg. 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 $c_{ij} \leftarrow F_{ij}(\mathbf{b}_i)$, which is a function of its current bundle, to create a score c_{ij} (line 9). Again, $F_{ij}(\mathbf{b}_i)$ is implicitly a function of the assignment constraints $\mathbf{G}(\mathbf{x}, \boldsymbol{\tau}) \leq \mathbf{d}$ posed in the problem formulation (Eq. 1) and if a proposed assignment will not satisfy the constraints required by Eq. 1, $F_{ij}(\mathbf{b}_i)$ 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).

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:      $\mathbf{b}_i \leftarrow \emptyset$
6:      while $|\mathbf{b}_i| < L_t$ do
7:          $\mathcal{J} \leftarrow \{j \mid \bar{s}_{ij} \notin \mathcal{A}_i\}$
8:          for all $j \in \mathcal{J}$ do
9:              $c_{ij} \leftarrow F_{ij}(\mathbf{b}_i)$
10:             $\bar{c}_{ij} \leftarrow \min\{c_{ij}, \bar{c}_{ij_{|\mathbf{b}_i|}}\}$ (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:         $j^\star \leftarrow \underset{j \in \mathcal{J}}{\operatorname{argmax}}(c_{ij} \cdot h_{ij})$
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:             $\mathbf{b}_i \leftarrow \mathbf{b}_i \oplus \bar{s}_{ij^\star}$
17:             $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \bar{s}_{ij^\star}$
18:         else
19:             break
20:         end if
21:     end while
22:     return $(\mathbf{b}_i, \mathcal{A}_i)$
23: end procedure
```

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}, whose 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 also 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 $\mathcal{A}_i$.

Bid Warped Task Consensus (Alg 4: `BW-TC`) The purpose of this function is to allow agents to exchange task assignment information with neighboring agents. The code in this algorithm is run independently for each agent, but neighboring agents synchronize their `Broadcast` (line 3) and `ReceiveMessages` (line 4) steps.

Algorithm 4 BW-TC: Bid Warped Task Consensus

```
1: procedure BW-TC($\mathbf{b}_i^o, \mathbf{b}_i, \mathcal{A}_i, Q_i, \mathcal{Z}_i$)
2:     $Q_i, \mathcal{Z}_i \leftarrow$ BW-TC-UQBC $(Q_i, \mathcal{Z}_i, \mathbf{b}_i^o, \mathbf{b}_i)$
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($Q_i$,$\mathcal{Z}_i$,$\mathbf{b}_i^o$,$\mathbf{b}_i$)
2:     for all $\{\bar{s}_{ij}^o \mid \bar{s}_{ij}^o \in \mathbf{b}_i^o,\ \bar{s}_{ij}^o \notin \mathbf{b}_i\}$ do
3:         $Q_i \leftarrow Q_i \cup \langle$Dropbid$(\bar{s}_{ij}^o), t_{now}\rangle$
4:         $\mathcal{Z}_i(i,j) \leftarrow t_{now}$
5:     end for
6:     for all $\{\bar{s}_{ij} \mid \bar{s}_{ij} \in \mathbf{b}_i,\ \bar{s}_{ij} \notin \mathbf{b}_i^o\}$ 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 time stamp 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 $\mathbf{b}_i^o$ and its new bundle $\mathbf{b}_i$. The details of this procedure are defined as Alg. 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 *dropbids*, 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 note 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 Alg. 6.

Bid Warped Task Consensus Update Queue with Bundle Changes (Alg 5: `BW-TC-UQBC`) The purpose of this function is to update the broadcast queue Q_i and local time stamp matrix $\mathcal{Z}_i$ with the local changes made to agent i's bundle $\mathbf{b}_i$ in the bundle building phase (Alg. 3).

1. This algorithm first searches over all bids $\bar{s}^o_{ij}$ that were in the old bundle $\mathbf{b}^o_i$ but not in the new bundle $\mathbf{b}_i$ (line 2). A dropbid is then created for all of these bids ($\texttt{Dropbid}(\bar{s}^o_{ij})$) and added to the broadcast queue Q_i (line 3). `Dropbid` is defined as a bid that signifies the removal of a previously placed bid. The function $\texttt{Dropbid}(s)$ returns a dropbid corresponding to bid s. Correspondingly, the time stamp 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 $\mathbf{b}_i$ but not in the old bundle $\mathbf{b}^o_i$ (line 6). Each of these new bids $\bar{s}_{ij}$ (with a time stamp) are added to the broadcast queue Q_i (line 7) and the local time stamp matrix $\mathcal{Z}_i$ (line 8) with the current time t_{now} plus a small extra value called δ. Its important that the time stamps 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.

Bid Warped Task Consensus Process Received Messages (Alg. 6: `BW-TC-PRM`) The purpose of this algorithm 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 time stamp 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 dropbids (lines 2-3). If the dropbid is new $\mathcal{Z}_i(i_m, j_m) < t_m$ (line 4) , update the time stamp matrix (line 5), add the dropbid 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 dropbid (lines 13, 14). Again, if it is a new bid (line 15), update the time stamp 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 time stamp 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 time stamp that is stored as $\mathcal{Z}_i(i', j_m)$ (line 27).

5.4 Comparison to Previous Work

The task consensus phase presented here is differs from the one that was presented in [40]. 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

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 Dropbid 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 Dropbid 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 $\bar{s}_{i_m j_m} > \bar{s}'_{i' j_m}$ 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
```

networks that change topology on the time scale 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 [40] 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 [40].

6 Convergence and Performance Characteristics of BW-CBBA

This section proves the two main theoretical results of the paper: 1) BW-CBBA converges to a team-wide consistent solution in at most $2N_t\mathcal{D}$ iterations and 2) BW-CBBA achieves non-trivial 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 $\mathbf{b}_i$), 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 [40] for 2 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), and 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 [40] when the internal score function F is submodular and monotonic.

6.1 Convergence Guarantee

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

$$\bar{s}_{ij_k} \succ \bar{s}_{ij_{k+1}} \qquad\qquad \forall k \in \{1, \dots, |\mathbf{b}_i| - 1\}$$

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.

2. $\bar{c}_{ij_k} = \bar{c}_{ij_{k+1}}$ but since $(k < k+1)$, $\bar{s}_{ij_k}$ is located earlier in the bundle than $\bar{s}_{ij_{k+1}}$. From Eq. 15 of Def. 1, $\bar{s}_{ij_k} \succ \bar{s}_{ij_{k+1}}$.

□

In order to reduce the notation clutter, all subsequent over bars will be removed, but all bids discussed in this convergence guarantee section 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 changes its local bid space $\mathcal{A}_i$ (Alg. 6). An overbid can also be a **dropbid** which was previously defined as a bid message specifying the removal of a previous bid. Dropbids are defined to have the same relative size of the bid they correspond to dropping (via Alg. 6, line 8) where the size of the bid is defined in Def. 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 $s_{ij} \in \mathbf{b}_i$ s.t.*

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

will remain in agent i's bundle $\mathbf{b}_i$ 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 $\exists s^{\ominus}_{ij^{\ominus}} \in \mathbf{b}_i$ s.t. $s^{\ominus}_{ij^{\ominus}} \prec \sigma_{i_\sigma j_\sigma}(S')$ and $\exists s_{ij} \in \mathbf{b}_i$ s.t. $s_{ij} \succ \sigma_{i_\sigma j_\sigma}(S')$ 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} can not 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 $s^{\ominus}_{ij^{\ominus}}$) and thus without loss of generality, s_{ij} will not depend on any bids in its own bundle $\mathbf{b}_i$ 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 $h_{ij} \leftarrow \prod_{s_{i'j} \in \mathcal{A}_i} \mathbb{I}(c_{ij} > c_{i'j})$ (Alg. 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 $s^{\star}_{i^{\star}j^{\star}}$ (Alg. 3, line 14) will return identical results for all bids $s^{\star}_{ij^{\star}} \succ \sigma_{i_\sigma j_\sigma}(S')$. □

A note about Thm. 1 is that it depends on bid warping to ensure Lemma 1. If bundles are not monotonic then $\exists i, s_{ij} \in \mathbf{b}_i, s^{\ominus}_{ij^{\ominus}} \in \mathbf{b}_i$ s.t. $s_{ij} \succ s^{\ominus}_{ij^{\ominus}}$ 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 is what bid warping prevents.

Theorem 2. *The largest new*[2] *bid $s^\star_{ij^\star}$ (Alg. 3, line 14) that can be added to agent i's bundle $\mathbf{b}_i$ 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 Thm. 1, all bids s_{ij} in agent i's bundle $\mathbf{b}_i$, s.t. $s_{ij} \succ \sigma_{i_\sigma j_\sigma}(S')$ will be preserved. For this proof, construct a new bundle $\mathbf{b}^+_i$ starting with all old elements $s^o_{ij} \in \mathbf{b}_i$ s.t. $s^o_{ij} \succ \sigma_{i_\sigma j_\sigma}(S')$. The strategy for the rest of this proof will be to show that the next bid $s^\star_{ij^\star}$ added to $\mathbf{b}^+_i$ 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 $\mathbf{b}_i$ that is less than $\sigma_{i_\sigma j_\sigma}(S')$. If no bids in $\mathbf{b}_i$ are smaller than $\sigma_{i_\sigma j_\sigma}(S')$, treat $s^\ominus_{i^\ominus j^\ominus}$ as an empty bid of score 0 (all bids with positive score 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 $\mathbf{b}^+_i$.

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} (Alg. 3 line 11) changes from a 0 to a 1. This can only occur if $s'_{i'j'}$ is specifically a *dropbid*, and thus removes a bid on task j' in agent i's local bid space $\mathcal{A}_i$ (Alg. 6, line 8). This is because only a dropbid can decrease the winning scores in $\mathcal{A}_i$. All other overbids will only increase the winning score in the local bid space. If Alg. 3 line 11 had previously been returning 0 (before the dropbid arrived), it means that the previous winning value on task j' (before the dropbid) $c'_{i'j'} > F_{ij'}(\mathbf{b}^+_i)$ (otherwise $h_{ij'}$ would have returned a 1). 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

$$c^\star_{ij^\star} = \max(F_{ij'}(\mathbf{b}^+_i), c^\ominus_{ij^\ominus})$$

and therefore from Def. 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 $\sigma_{i_\sigma j_\sigma}(S')$. Since $s^\star_{ij^\star}$ 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 BW-CBBA agrees on a globally consistent bidspace $\mathcal{A}$ in at most $2N_t\mathcal{D}$ 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 $\mathcal{A}$.

[2] "New" here refers to a bid that was not included in the bundle of the previous bundle building iteration

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 BW-CBBA, the algorithm starts with a bundle building phase (Alg. 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 mid-mission replan, $\mathcal{A}_i$ will be the last known estimate of the team assignment (which may be outdated). As defined in Alg. 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 $\mathbf{b}_i = \{\}$ (line 5). After the first iteration of bundle building completes (Alg 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 $\mathbf{b}_{i'}$ of agent i'.) More precisely, $\exists i, s'_{i'j'} \in \mathcal{A}_i$ s.t. $\forall i, \forall s_{ij} \in \mathbf{b}_i, s'_{i'j'} \succ s_{ij}$. 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 dropbid removing the assignment of $s'_{i'j'}$ (constructed via Alg. 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 $\mathbf{b}_{i'}$ of some agent i' s.t. $\forall i, \forall s_{ij} \in \mathbf{b}_i,\ s.t.\ s_{ij} \neq s'_{i'j'}, s'_{i'j'} \succ s_{ij}$. From Thm. 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 Thm. 1, $s'_{i'j'}$ will remain in the bundle of agent i' forever because, since 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 $s'_{i'j'}$ and 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 $\forall i \nexists s_{ij} \in \mathbf{b}_i$ s.t. $s_{ij} \notin \mathcal{A}$ (the algorithm has converged) or $\exists s'_{i'j'} \in \mathbf{b}_{i'}, s'_{i'j'} \notin \mathcal{A}$ s.t. $\forall i, \forall s_{ij} \in \mathbf{b}_i, s_{ij} \notin \mathcal{A}, s'_{i'j'} \neq s_{ij}, s'_{i'j'} \succ s_{ij}$.

In this step three scenarios can arise:

1. There are no bids in the network that are not already in $\mathcal{A}$ or more formally, $\forall i \nexists s_{ij} \in \mathbf{b}_i$ s.t. $s_{ij} \notin \mathcal{A}$. 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 $\mathbf{b}_{i'}$ that is smaller than s_{ij}. More formally, the scenario arises if $\exists s_{ij} \in \mathcal{A}$ s.t. $\exists s'_{i'j} \in \mathbf{b}_{i'}$ s.t. $i \neq i'$. 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 dropbid $s^{\ominus}_{i'j^{\ominus}}$ is larger than all other bids not yet in $\mathcal{A}$ (if $\forall i, \forall s_{ij} \in \mathbf{b}_i, s_{ij} \notin \mathcal{A}, s^{\ominus}_{i'j^{\ominus}} \neq s_{ij}$, $s^{\ominus}_{i'j^{\ominus}} \succ s_{ij}$), then the algorithm requires an additional $\mathcal{D}$ iterations to allow for this dropbid 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 Thm. 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 dropbid 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 below.

3. There exists a bid $s'_{i'j'}$ in some agent i''s bundle $\mathbf{b}_{i'}$ that is larger than all other bids s_{ij} in every other agent's bid spaces that is not currently located in $\mathcal{A}$ ($\exists s'_{i'j'} \in \mathbf{b}_{i'}$ s.t. $s'_{i'j'} \notin \mathcal{A}$ and $\forall i, \forall s_{ij} \in \mathcal{A}_i, \forall s_{ij} \neq s'_{i'j'}, s_{ij} \notin \mathcal{A}, s'_{i'j'} \succ s_{ij}$). Since 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 scenario 2 above), 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 Thm. 1). Additionally, since i' will never receive an outbid message greater than $s'_{i'j'}$ (because no other agents can bid higher (Thm. 2), $s'_{i'j'}$ will stay in the bundle of agent i forever (Thm. 1). Therefore, $\mathcal{A} \leftarrow \mathcal{A} \cup s'_{i'j'} \implies |\mathcal{A}| = n+1$ in 1 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 ($2\mathcal{D} - 1$ iterations from scenario 2, and 1 iteration from scenario 3 fro each task). This also means that every individual agent will agree on the full bid space in at most $2N_t\mathcal{D}$, and thus BW-CBBA has converged.

□

6.2 Performance Guarantee

This section will define when non-trivial 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 [48]. 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 $\mathcal{A}$. Further, define the notation $\mathcal{A}_F^{BW}$ to be the bid space returned using objective function F with BW-CBBA. Similarly, define $\mathcal{A}_F^{\star}$ 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 $\mathcal{A}_F^{BW}$ where $s_{i_k j_k}$ is defined to be the kth element added to $\mathcal{A}_F^{BW}$ (which was constructed by agent i_k on task j_k) using the global bid space construction procedure from Thm. 3.

The form for the following proof will be to compare the optimal allocation $\mathcal{A}_F^{\star}$ evaluated on the desired non-submodular objective function $F(\mathcal{A}_F^{\star})$, to the optimal allocation $\mathcal{A}_{F_k}^{\star}$ 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 4. *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*

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

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

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

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

Definition 5. *Define a submodular lower bound function* H *to the desired mission objective function* F *as*

$$\begin{aligned} H(\emptyset) &= F(\emptyset) \\ H(\mathcal{A} \cup s) - H(\mathcal{A}) &= \min_{\mathcal{A}' \subseteq \mathcal{A}} (F(\mathcal{A}' \cup s) - F(\mathcal{A}'))\ \forall s, \mathcal{A}. \end{aligned}$$

Definition 6. *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 non-monotonic functions. If the desired mission objective function* F *is actually submodular then* $\epsilon = 0$ *and* $F = H$*.*

Combining Def. 6 and Def. 5 provides a submodular upper and lower bound on F for all bid spaces $\mathcal{A}$ as

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

Definition 7. *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 Def. 5,*

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

Proof. Two cases are possible, either

1. $s_{i_k j_k}$ does not change its value when warped and thus its warped value is the incremental score w.r.t. objective function F

$$\begin{aligned}\bar{c}_{i_k j_k} &= F(\mathcal{A}_{k-1} \cup s_{i_k j_k}) - F(\mathcal{A}_{k-1}) \\ &\overset{\text{Def. 5}}{\geq} H(\mathcal{A}_{k-1} \cup s_{i_k j_k}) - H(\mathcal{A}_{k-1})\end{aligned}$$

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,

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

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 (Def. 5), a minimum over a larger set will always be smaller and 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 Def. 5 and ϵ as defined in Def. 6 and the mission objective function F is monotonic, then a provable performance bound between the BW-CBBA allocation $\mathcal{A}_F^{BW}$ and the optimal allocation $\mathcal{A}_F^{\star}$ exists as*

$$F(\mathcal{A}_F^{\star}) \leq (2+\epsilon)\, F\left(\mathcal{A}_F^{BW}\right). \tag{23}$$

Proof. In order to simplify the notation of the following proof, a few notational substitutions will be made:

1. $s_k^{\star} \leftarrow s_{i_k^\star j_k}^{F_{k-1}}$ 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. $s_k \leftarrow s_{i_k j_k}$ to represent the kth bid from the BW-CBBA bid space $\mathcal{A}_F^{BW}$,

3. $\mathcal{A}^\star_\ominus \leftarrow \mathcal{A}^\star_{F_{k-1}} \setminus s^\star_k$ to represent the optimal bid space w.r.t. F_{k-1} without the bid $s^\star_k$ that was made on task j_k.

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

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

Then using the definition of F_k from Def 4, the right hand side can be expanded as

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

which can then be expanded as 3 terms

$$\begin{aligned} F_k(\mathcal{A}^\star_\ominus) = & F_{k-1}\left(\mathcal{A}^\star_{F_{k-1}}\right) - && \text{(25a)}\\ & \left(F_{k-1}\left(\mathcal{A}^\star_{F_{k-1}}\right) - F_{k-1}\left(\mathcal{A}^\star_\ominus\right)\right) - && \text{(25b)}\\ & \left(F_{k-1}\left(s_k\right) - \left(F_{k-1}\left(\mathcal{A}^\star_\ominus \cup s_k\right) - F_{k-1}\left(\mathcal{A}^\star_\ominus\right)\right)\right). && \text{(25c)} \end{aligned}$$

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 w.r.t 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 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 BW-CBBA and c_k is the un-warped 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. Since 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 (Def 5) and ϵ (Def 6) from Eq. 18 and the equivalence of the two forms of F_k from Def. 4 implies that,

$$(1+\epsilon)\bar{c}_{i_k^\star j_k} \geq \left(F_{k-1}\left(\mathcal{A}^\star_{F_{k-1}}\right) - F_{k-1}\left(\mathcal{A}^\star_{\ominus}\right)\right). \tag{29}$$

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

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

The next objective is to bound the term defined as Eq. 25c. From the definition provided as Def. 7, 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,

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

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

$$F(\mathcal{A}^\star_F) \leq (2+\epsilon)\, F\left(\mathcal{A}^{BW}_F\right). \tag{32}$$

□

6.3 Algorithmic Comparison Results

This section provides performance and convergence comparisons between several global information consistency assumption (GICA) algorithms and local information consistency assumption (LICA) algorithms . The results will show that LICA algorithms can significantly reduce algorithmic convergence time over competing GICA algorithms. Additionally the bid warping approach described [4] can significantly improve the performance of LICA algorithms. In fact, in almost all of the domains tested, BW-CBBA actually returns the same allocation as GICA algorithms.

6.3.1 Non-submodular fuel penalty: 2 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:

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

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 $d_i(\mathbf{b}_i)$ by agent i for its assigned group of tasks $\mathbf{b}_i$. Figure 8 visually compares the planned paths for a 2 agent, 30 task mission, using the original

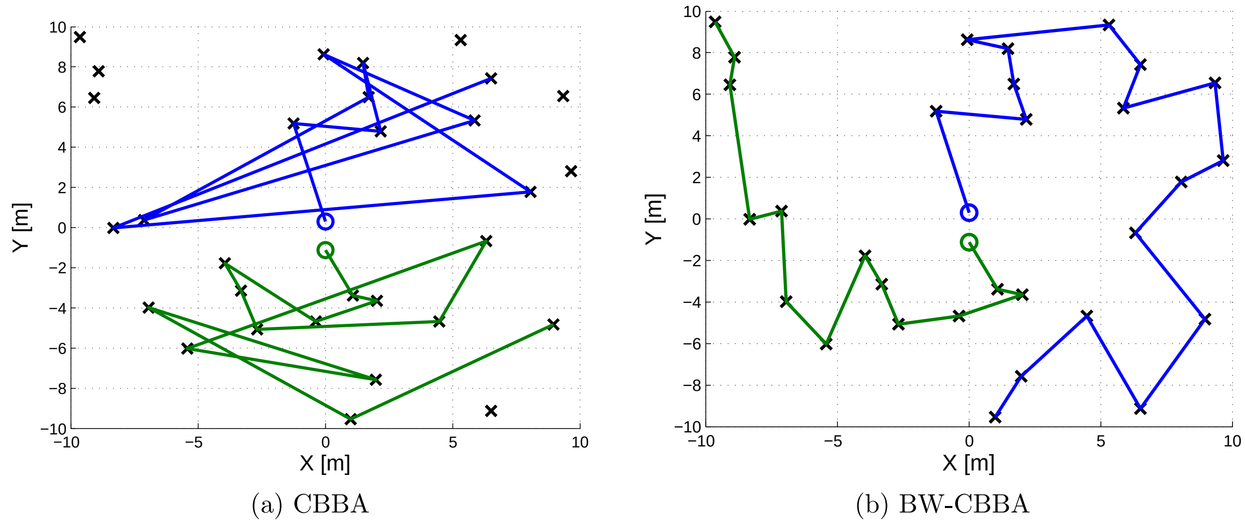


Figure 8: Comparison of planner performance for a 2 agent, 30 task mission. Fig (a) shows the planned paths using the original CBBA algorithm with a submodular heuristic score function. Fig (b) shows the planned paths for the same scenario using BW-CBBA as proposed in [4].

baseline CBBA algorithm with a submodular approximate score function (a) and BW-CBBA augmented to utilize the true non-submodular score function (b). The numerical values used for this experiment were a reward $R = 100$ and a fuel penalty $f_i = 10$. As was introduced in [4], one heuristic approach to ensure submodularity within the original CBBA framework involves approximating the cost in Equation (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 (Figure 8(a)). Conversely, figure 8(b) demonstrates how 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.

6.3.2 Non-submodular fuel penalty: Monte Carlo results

This experiment provides Monte Carlo results comparing the performance in various mission scenarios for 5 different algorithms. These displayed scenarios 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 and (2) by utilizing BW-CBBA the score performance gap of using a LICA algorithm is negligible in practice. The environment for these Monte Carlo tests placing tasks and agents at random locations in a 2 dimensional rectangle of dimensions roughly 34 by 12. (These seemingly random numbers correspond to the shape of our research flight volume). The mission predictions were run in continuous time where the speed of the agents was fixed at a maximum of .6, the time agents were required to pause to "complete" the tasks was set to 1 time unit, and all tasks expired after 100 time units. The objective function for the environment was identical to Equation (33) with a reward $R = 100$ and a fuel penalty $f_i = 10$. The agents have identical fuel penalties and speeds but this is not required by any algorithms used in this 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 Figure 9 is an average number over 500 Monte Carlo trials. The computational environment for these experiments were conducted with agents operating independently on separate computers communicating over a virtual communication network that was defined as a minimum spanning over the agents' randomized starting positions. This provides a level of realism the tested performance of the algorithms because the only information shared between the agents were the actual task bids shared during communication, and true iteration synchronization was required between the distributed agents. The five planners used in these tests are

1. Sequential auction. This is a GICA algorithm (as defined in [42]) which essentially involves the team incrementally building up a global bid space 1 task at a time.This requires the every member of the team communicating with all other agents for every single task assignment.

2. Implicit Coordination. The implicit coordination implementation in these experiments utilize a poor information environment because each agent independently optimizes its own objective function ignoring the contributions of other agents. The result gives the team-wide performance when no cooperation is used. [19, 21, 49]

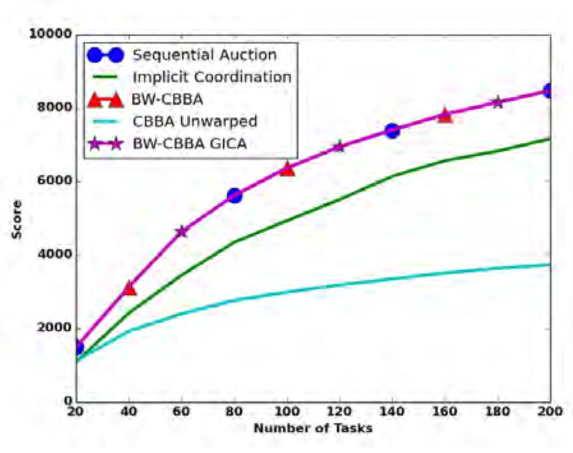
3. BW-CBBA (as defined in [4]) is the LICA contribution of this work.

4. CBBA Unwarped refers to the baseline solution described in [40] which uses an a priori approximate submodular function.

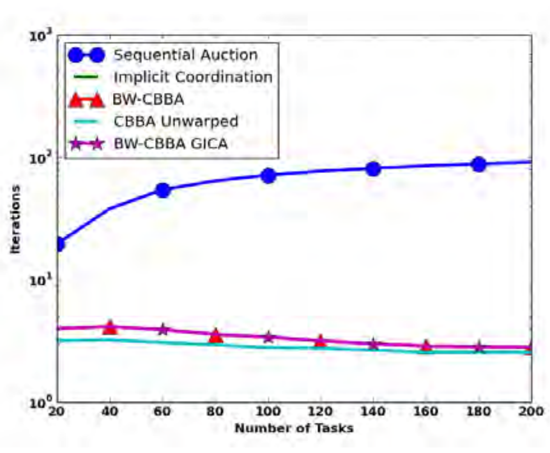
5. BW-CBBA with GICA, a variant of BW-CBBA implemented for these experiments, that ensures that every agent's local bid space is equivalent to the global bid space before every bundle building phase. This algorithm is essentially a bundle version of a sequential auction and thus approximates the fastest 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 is not considered in this particular work. Traditionally these approaches would be considered GICA algorithms because they require information about the entire fleet to guarantee convergence [19, 21, 49], but recent work that will be described later in this report looked at LICA algorithms that use information about other agents score functions [8].

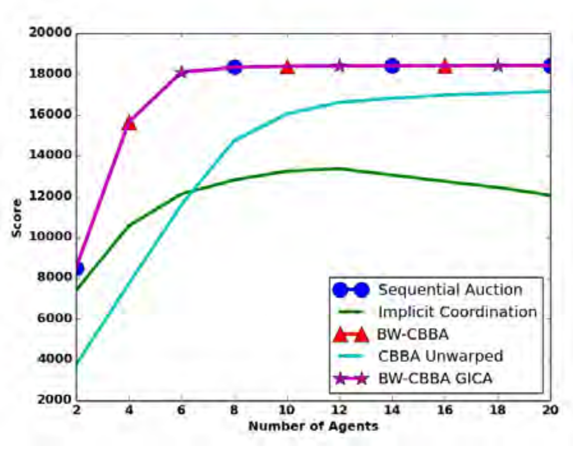
The experiments shown in figures 9 (a) and (b) are a 2 agent Monte Carlo run with 500 trials averaged for each data point. Figure (a) shows that BW-CBBA performs identically to both of the GICA algorithms. The implicit coordination technique performs reasonable well, because since the tasks expired at 200 seconds was not possible for both agents to service all of the tasks and thus there was relatively little overlap in desired assignments



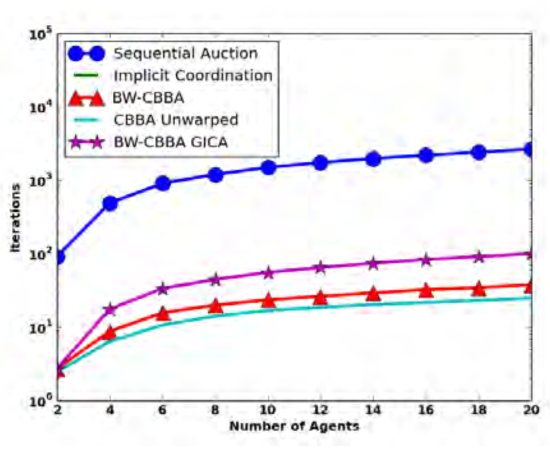
(a) 2 Agent Example (Score)



(b) 2 Agent Example (Iterations)



(c) 200 Task Example (Score)



(d) 200 Task Example (Iterations)

Figure 9: (a) 2 Agent Monte Carlo run with 500 trials showing score as a function of the number of tasks. (b) 2 Agent Monte Carlo run with 500 trials showing number of iterations to convergence as a function of the number of tasks. (c) 200 task Monte Carlo run with 500 trials showing score as a function of the number of agents. (d) 200 task Monte Carlo run with 500 trials showing the number of iterations to convergence as a function of the number of agents

between the agents in this no coordination planner. Unwarped CBBA actually performs quite poorly, especially for high numbers of tasks because its objective function cannot capture the super-modular coupling between servicing tasks that are near each other. Figure 9 (b) highlights the convergence times for each of these algorithms. The first remarkable thing about this figure is the large number of iterations required for convergence using the sequential auction algorithm. Even though there are only 2 agents and thus the network diameter is 1, it takes a full iteration for the assignment of every single task. 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 2 agent network is fully connected providing the agents using BW-CBBA to have consistent local bid spaces after every iteration, and thus the algorithmic execution of BW-CBBA and BW-CBBA GICA is identical. These approaches take less than 1 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 true coupled objective function. Since there is no explicit coordination with implicit coordination it always "converges" in 1 iteration. The take away from figures (a) and (b) is that even with 2 agents, BW-CBBA outperforms the unwarped CBBA significantly in score performance for a small penalty in an increased number of iterations it takes to converge. Additionally, its performance is identical to the tested GICA planners.

The experiments shown in Figures 9 (c) and (d) are Monte Carlo runs with 200 task environment and 500 trials averaged to create each data point. Figure (c) shows identical performance between BW-CBBA and the GICA algorithms. Additionally, it shows a significant performance gap between unwarped CBBA and the implicit coordination approach especially for smaller team sizes. When the team size reaches 8 agents and above, BW-CBBA and the GICA algorithms are able to service all of the tasks efficiently under the 100 second 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, and 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 9 (d) 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. 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. BW-CBBA a LOCA algorithm further reduces the number of iterations to convergence below BW-CBBA GICA, in fact for the largest problem sizes shown 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

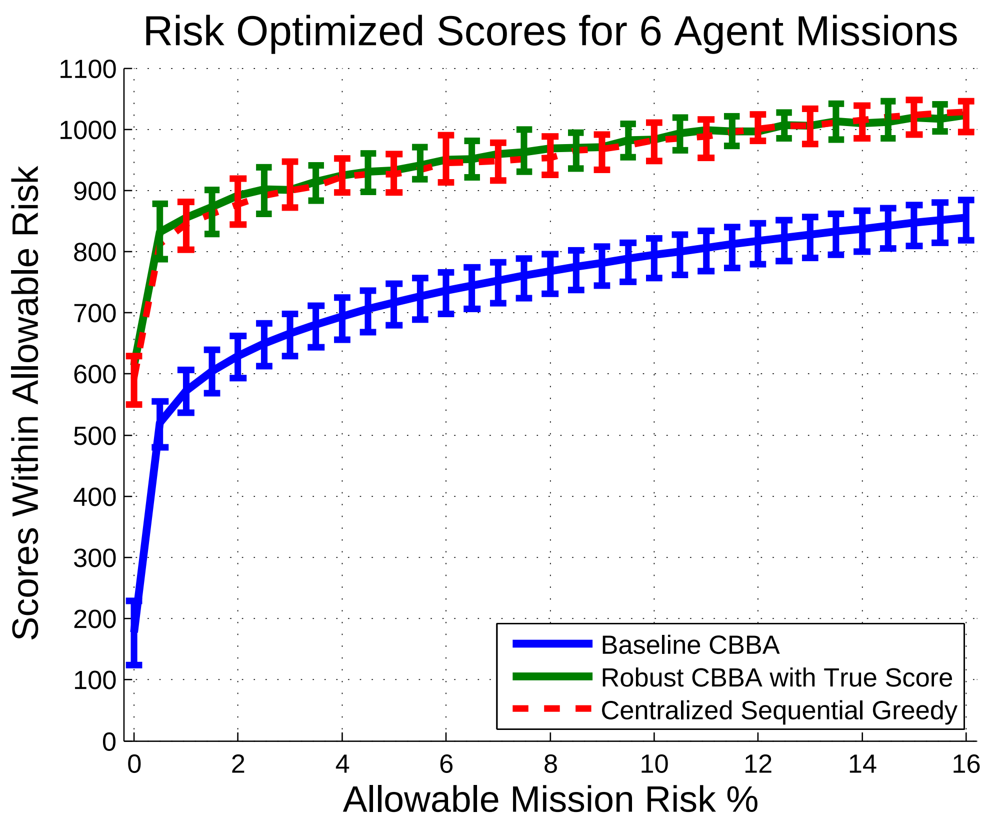


Figure 10: Monte Carlo results for a 6 agent, 60 task, time-critical mission, with uncertain travel times and task durations. Results show risk-optimized robust mission scores for the different planning approaches.

resolution is more efficiently conducted using local communication. CBBA unwarped converges marginally faster than even BW-CBBA but again this is at the expense of significant degradation in score performance. The take away from figures 9 is that 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 figures 9 that involves the on-board 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 figure 9, this agent computation times were negligible compared to the required infrastructure to synchronize communication between the decentralized agents.

6.3.3 Stochastic objective functions

The third scenario considered is meant to broaden the applicability of bid warping to a less straightforward introduction of non-submodularity. In this environment, agents must now service tasks with uncertainty in the planning parameters and task rewards that are heterogeneously time-critical. In particular, the agents have uncertain velocities and service times for tasks, although a probability distribution of the possible values are known a priori (lognormal). In this stochastic setting, submodularity can be broken both because of sampling affects (as introduced in Example 4), and because of the stochastic metrics implicitly couple the effects of parameter uncertainty, and therefore couple task scores. This leads to dependencies between tasks which can cause non-submodularity in the score function. This type of coupling and non-submodularity is typically non-trivial, and designing a submodular score function to fix it is difficult. This example demonstrates that BW-CBBA can be successfully used for distributed stochastic planning. The algorithm uses a "repeatable stochastic score function" which reuses samples associated with planning parameters, making the problem look like a higher dimensional deterministic problem to the planner. It is worth noting that non-submodularity in stochastic settings is usually an issue even when the mean score function is submodular, however, for consistency, this example uses the non-submodular form for the mean score function Equation (33) (with non-submodularity caused by fuel penalties). Figure 10 presents Monte Carlo results for a 6 agent, 60 task, time-critical mission, with uncertain travel times and task durations. The plot shows the risk-optimized mission scores for a centralized robust sequential greedy planner, the distributed robust BW-CBBA approach [1], and a baseline deterministic CBBA. Once again the new stochastic CBBA approach achieves similar performance to the centralized planner and clearly outperforms the baseline unwarped CBBA approach.

6.4 Analysis of Algorithmic Performance

This section provides further insight into the interpretation of the performance bounds of BW-CBBA. Performance guarantees are provided in Thm. 4 when the score functions can be approximated with a submodular function H and distance parameter ϵ. The follow section discusses what kind of performance can be expected when the conditions required for Thm. 4 are not met.

An optimal upper bound for the score function defined as Eq. 33 in the results section using the can be defined as

$$\sum_{j=1}^{N_t} \max\left(\max_i \left(R_{ij} - f_i d_{ij}\right), \max_{i,j'} \left(R_{ij} - f_i d_{j'j}\right), 0\right), \tag{34}$$

where d_{ij} is the distance from agent i to task j. This bound essentially finds the maximum possible reward achievable on each task, without the constraint of connected agent trajectories by agents. This upperbound will almost always be loose but nevertheless will give some insight into the performance of the algorithms in Sec 6.3.2.

1. **Approximately Submodular** With these objective functions there will be very little difference between F and H and 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 very near optimal.

2. **Non-submodular with good local optima** This is the category that the simulations

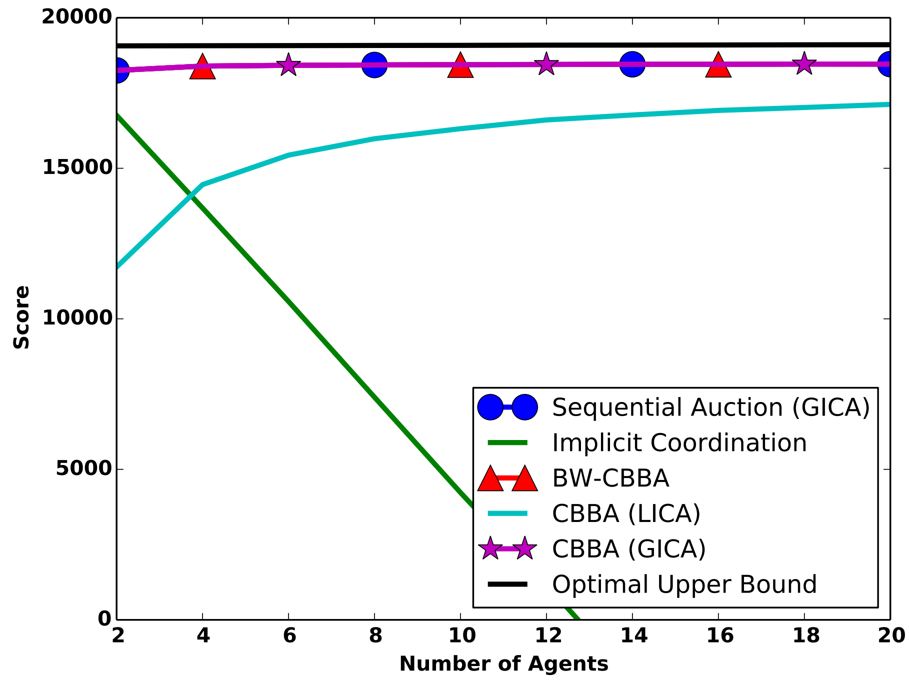


Figure 11: Comparison of mission score in 500 Monte-Carlo trials of a 200 task mission with a varying number of agents.

in Sec. 6.3 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. 9 were specifically designed to highlight how BW-CBBA could perform well in substantially non-submodular environments, and as a result there does not exist a non-trivial ϵ and H for the objective function defined as Eq. 33. Figure 11 presents a slight modification to the problem statement presented in Sec 6.3.2. In order to create a reasonable optimal upper bound, the problem was simplified by removing the task time out constraints. As can be seen from the optimal upper bound in Fig. 11, all of the GICA approaches and BW-CBBA perform well (as the black line is only an upper bound on optimal). 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 team-wide performance. This occurs in domains like those presented in Sec. 6.3.2 where tasks are placed randomly in a 2-dimensional grid and 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. **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 BW-CBBA can perform arbitrarily poorly. In these domains there can be sufficient objective function coupling such that no sequential greedy algorithm could 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, not just 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.

7 Hybrid Information and Plan Consensus (HIPC)

The first incremental step at solving the problem mentioned so far was just bootstrapping a plan consensus algorithm with an initial implicit coordination allocation. The first shot at this was called CBBA with bootstrapping. The main structure of this algorithm was to create an implicitly coordinated allocation using what ever information was available amongst the agents. If this was computed after global situational awareness consensus, the algorithm would converge immediately. If there were conflicts in the assignment, or high value tasks had gone unassigned, a plan consensus algorithm (CBBA) [40] was run using the implicit coordination solution as the initial bundle on board each agent.

The strength of this approach was that if there existed highly coupled behavoirs what were obvious to the agents even with partial information, the initial implicit coordination planner would find them, and this assignment would end up sticking. However, all it could take is 1 small piece of missing information to cause a cascading failure in the initial assignment, and the algorithm would completely revert to a plan consensus algorithm (CBBA). This did not achieve the goal of creating a planner that can really utilize this partial information agents have about each other to produce highly coupled plans.

The intuitive solution to bootstrapping's failures is to run both implicit coordination and plan consensus at every iteration of the planner. This is in fact the premise behind HIPC. The resulting algorithm is no longer as simple as appending together two different algorithms and running them in series. By combining both approaches at every iteration of the algorithm, the predictive power of implicit coordination is achieved, as well as introducing the conflict resolution power of plan consensus, at the cost of having each piece violate the others assumptions..

Recall that one of the properties of plan consensus algorithms that they trivially trusted the past bids of all other agents. They could be sure that high value tasks wouldn't go unassigned because these algorithms communicate which tasks are assigned by definition.

This is no longer the case in general because if an agents situational awareness is poor enough in a combined approach, they could be overestimating another agent's performance on a task and therefore it could potentially go unassigned all together. HIPC resolves this by creating a new algorithmic behavior of *understanding* when an agent is likely overestimating the performance of a teammate. In this case, agents learn to stop trusting that their neighbors will want to service a task and instead bid on it themselves.

The other useful feature of this new hybrid approach is that it also fixes a previous failure mode of pure implicit coordination approaches. This failure mode arose when agents would underestimate the performance of their teammates. This would lead to final allocations that would have conflicting assignments. Since plan consensus explicitly resolves conflicting assignments, this failure mode can be easily resolved by normal plan consensus mechanisms. The result is that in the creation of this hybrid algorithm, the performance loss from the lack of cooperation prediction of plan consensus algorithms is mitigated using implicit coordination, while the performance loss associated with over and underestimating performance with implicit coordination can be mitigated with plan consensus.

Theoretically this formulation is appealing but in practice some properties of hybrid algorithms are even nicer than they may appear. From the structure of this construction, asymmetry in information and communication is handled naturally. The assumptions from construction is that all extra information is useful, and thus asymmetries due to increasing the information of particular agents only increases the power of the allocation algorithm. Additionally, in environments where communication and information disperse over physical distances, its common for nearby agents to know the most about each other while little is known about teammates far away. In practice, distance is often a penalty in reward functions so nearby agents tend to be the ones that collaborate most often. Since cooperation is between nearby agents, and information consensus is strongest amongst nearby agents, the predictive power is actually stronger among the pairings that actually need it. For example, in a very large team, an agent may only have information about 1 percent of the vehicles in a fleet, which happen to be the ones in its line of sight. However, that agent is likely to only to need intense cooperation with other vehicles in his vicinity, so this 1 percent is all the information he actually needs to form highly coupled plans. Conversely, pure implicit coordination approaches that only has 1 percent of the global information state onboard each agent would likely perform poorly.

The rest of this document will discuss the details of algorithms that accomplish this hybrid planning. The above sections have hopefully motivated the theoretical need for these algorithms, and why they are not trivial extensions of an already existing expansive literature.

7.1 HIPC Notation

The Hybrid Information and Plan Consensus (HIPC) algorithm is a distributed algorithm that provides task assignments for multi-agent, multi-task allocation problems. The algorithmic structure of HIPC is an iterative, 2 phase algorithm. These two phases are: a *local bid space creation* phase where each agent generates a personal allocation of tasks (possibly using situational awareness of other agents in the team), and a *task consensus* phase where conflicting assignments are identified and resolved through local communication between adjacent agents. These two phases are repeated until the algorithm converges. To further explain the relevant details of the algorithm, some notation will be formalized.

Bid A bid is represented as a triple: $s_{ij} = \langle i, j, c \rangle$, where i is the bidding agent's index, j is the bid's task id, and c represents the bid score.

Bundle A bundle is an ordered data structure internal to each agent i, $\mathbf{b}_i = \{s_{ij_1}, \ldots, s_{ij_n}\}$ that consists of all n of its current bids. When new bids are made, they are appended to the end of the bundle, thus the order in the bundle reflects the relative age of each bid and thus the *dependency structure* of the bids, where later bids scored depend on the assignment of all earlier bids in the bundle.

Bid Space 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 the index of the last element in the bid space. This set contains a globally consistent set of the current winning bids in the fleet.

Local Bid Space 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 communication phase, but in general, the geometry of agents in the network may lead to information propagation latencies and thus non-identical local bid spaces.

Network Diameter The network diameter, $\mathcal{D}$, is defined as the communication distance between the furthest agents in the communication network. More formally, define the communication distance between any pair of agents i and i' to be $\mathcal{D}_{i \to i'}$. Define $\mathcal{D} = \max_{i,i'} \mathcal{D}_{i \to i'}$ to be the maximum communication distance over all agent pairs i and i'.

Neighborhood The neighborhood, $\mathcal{N}_i$, of an agent i is defined as the set of agents that agent i has situational awareness over. Similarly an agent i's exact neighborhood $\bar{\mathcal{N}_i}$ is the set of agents that agent i has *perfect* situational awareness over. By definition, an agent's perfect neighborhood always includes itself ($i \in \bar{\mathcal{N}_i}$) and an agent's perfect neighborhood is always a subset of its full neighborhood ($\bar{\mathcal{N}_i} \subseteq \mathcal{N}_i$). If an agent $i' \in \mathcal{N}_i$, then agent i expects that it can predict the objective values for agent i'. This does not mean that their task allocations will be trivially identical even when perfect information is known because neither their neighborhoods ($\mathcal{N}_i \neq \mathcal{N}_{i'}$) nor their local bid spaces ($\mathcal{A}_i \neq \mathcal{A}_{i'}$) will be identical in general.

7.2 HIPC Algorithmic Description

The high level HIPC algorithmic description is given in Algorithm 7.

1. HIPC is a procedure run on-board each agent independently. HIPC is initialized with an initial bid space $\mathcal{A}_i^0$, an available task set $\mathcal{J}$, and a set of agents $\mathcal{N}_i$ that each agent i has situational awareness over (Line 1). Note that $\mathcal{N}_i$ will contain a set of agents $\bar{\mathcal{N}_i}$ that agent i knowingly has perfect situational awareness over.

Algorithm 7 HIPC for each agent i

```
 1: procedure HIPC($\mathcal{A}_i^0$,$\mathcal{J}$,$\mathcal{N}_i$)
 2:     set $\mathcal{A}_i = \mathcal{A}_i^0$, $\mathcal{A}_i' = \mathcal{A}_i^0$
 3:     $iterationNumber = 1$
 4:     while $convergenceCounter < 2\mathcal{D}$ do
 5:         $(\mathcal{A}_i, \mathcal{N}_i) = \text{CreateHIPCAssignment}(\mathcal{A}_i', \mathcal{N}_i)$
 6:         $\mathcal{A}_i' = \text{Consensus Phase}(\mathcal{A}_i', \mathcal{A}_i)$
 7:         if $\mathcal{A}_i' = \mathcal{A}_i$ then
 8:             $convergenceCounter = convergenceCounter + 1$
 9:         else
10:             $convergenceCounter = 0$
11:         end if
12:         $iterationNumber = iterationNumber + 1$
13:     end while
14: end procedure
```

2. Before each bid space construction operation, each agent checks for convergence. If $\mathcal{A}_i$ hasn't changed for two times the network diameter ($2\mathcal{D}$) number of iterations, then the algorithm has converged (Line 4).

3. The algorithm calls the `CreateHIPCAssignment` subroutine. The objective of this function is to take in the current local bid space $\mathcal{A}_i'$ and the neighborhood set $\mathcal{N}_i$ and compute an updated local bid space $\mathcal{A}_i$ and potentially an updated neighborhood set $\mathcal{N}_i$ (Line 5). The neighborhood set would only be updated if agent i decided that planning for some agent i' was hindering algorithmic performance and it dropped i' from $\mathcal{N}_i$. The exact details of this subroutine are outlined in Alg. 8.

4. The next algorithmic step involves running the consensus phase where each agent i, shares its personal bundle $\mathbf{b}_i$ with its network neighbors (Line 6). This consensus phase can be implemented identically to the one proposed in [40]. (If the network is static, other consensus protocols exist where each agent only needs to share the changes to $\mathbf{b}_i$ reducing overall communication. If the communication network is non-static during plan construction, the full $\mathbf{b}_i$ will need to be shared to retain the worst convergence bounds.)

5. The algorithmic convergence condition is checked (Line 7), if it is true then the $convergenceCounter$ is incremented by one (Line 8), if not $convergenceCounter$ is reset to zero (Line 10).

6. Lastly the iteration counter is incremented (Line 12) and the algorithm returns to Line 4.

7.2.1 Creating the HIPC Assignment

The following section describes Alg. 8 which dictates how the local bid space $\mathcal{A}$ and the neighborhood $\mathcal{N}_i$ are incrementally updated inside the HIPC algorithm.

Algorithm 8 Creating the HIPC Assignment i

```
1: procedure CreateHIPCAssignment($\mathcal{A}_i'$,$\mathcal{N}_i$)
2:     $\hat{\mathcal{N}_i} \leftarrow \emptyset$
3:     while $\hat{\mathcal{N}_i} \neq \mathcal{N}_i$ do
4:         $\hat{\mathcal{N}_i} \leftarrow \mathcal{N}_i$
5:         $\hat{\mathcal{A}}_i \leftarrow \mathcal{A}_i'$
6:         for all $s_{i'j} \in \hat{\mathcal{A}}_i$ s.t. $i' \in \bar{\mathcal{N}_i}$ do
7:             $\hat{\mathcal{A}}_i \leftarrow \hat{\mathcal{A}}_i \notin s_{i'j}$
8:         end for
9:         $\hat{\mathcal{A}}_i \leftarrow TAA(\hat{\mathcal{A}}_i, \mathcal{N}_i)$
10:        $\hat{\mathcal{N}_i} =$ CheckSAConsistency($\hat{\mathcal{A}}_i, \mathcal{A}_i', \mathcal{N}_i$)
11:    end while
12:    Return ($\hat{\mathcal{A}}_i, \hat{\mathcal{N}_i}$)
13: end procedure
```

1. The procedure takes in the output from consensus at the previous iteration $\mathcal{A}_i'$ and the current neighborhood set $\mathcal{N}_i$ (Line 1).

2. The first time the algorithm reaches $\hat{\mathcal{N}_i} \neq \mathcal{N}_i$, (Line 3), the while loop will trivially return true ($|\mathcal{N}_i| \geq 1$.) Each subsequent time through (Line 3) checks if the neighborhood set has changed, when it hasn't, the procedure returns.

3. Drop all tasks from the local bid space $\hat{\mathcal{A}}_i$ that belong to agents in known perfect situational awareness set $\bar{\mathcal{N}_i}$ (Lines 6-8).

4. Use the TAA to compute an updated local bid space, $\hat{\mathcal{A}}_i$. New bids are computed locally for every agent in the neighborhood set $\mathcal{N}_i$.(Line 9) The default behavior inside of the TAA function is that agents cannot remove any bids that they have already committed to, they can only bid on unassigned tasks, or outbid a current winner of a task. Note that all bids made by agents in $\bar{\mathcal{N}_i}$ are removed before entering TAA, so the bundles for these agents are built up from scratch.

5. Update neighborhood set $\hat{\mathcal{N}_i}$ from the output of `CheckSAConsistency` (Line 10).

6. Return new local bid space $\hat{\mathcal{A}}_i$ and new neighborhood set $\hat{\mathcal{N}_i}$ (Line 12).

7.2.2 Checking Situational Awareness Consistency

The following section describes Alg. 9 which describes how agents i' are removed from agent i's neighborhood $\mathcal{N}_i$.

1. The procedure takes in the new predicted bid space $\mathcal{A}_i$ the output from consensus at the previous iteration $\mathcal{A}_i'$ and the current neighborhood set $\mathcal{N}_i$ (Line 1).

2. Iterate over all agents that i has imperfect SA over (Line 2).

3. Find the max bid predicted for agent i' that is in the new bid space $\mathcal{A}_i$ but not in the old one $\mathcal{A}_i'$ (Line 3)

Algorithm 9 Checking SA consistency with assignments i

```
 1: procedure CheckSAconsistency($\mathcal{A}_i, \mathcal{A}'_i, \mathcal{N}_i$)
 2:     for all $i' \in \mathcal{N}_i \setminus \bar{\mathcal{N}}_i$ do
 3:         $\bar{s}_{i'j} = \max(s_{i'j})$ s.t. $s_{i'j} \in \mathcal{A}_i$ , $s_{i'j} \notin \mathcal{A}'_i$
 4:         $\gamma \leftarrow \langle \cdot, \bar{s}_{i'j} \rangle$
 5:         if $\gamma \in \Gamma_{i'}$ then
 6:             if $z + \mathcal{D}_{i \to i'} + \mathcal{D}_{i' \to i} < iterationNumber$ then
 7:                 $\mathcal{N}_i \leftarrow \mathcal{N}_i \setminus i'$
 8:                 continue
 9:             end if
10:         else
11:             $\Gamma_{i'} = \Gamma_{i'} \cup \langle iterationNumber, \bar{s}_{i'j} \rangle$
12:         end if
13:     end for
14:     Return ($\mathcal{N}_i$)
15: end procedure
```

4. Check if this bid is in agent i's previous overbid list for agent i', defined as $\Gamma_{i'}$ (Line 5). $\Gamma_{i'}$ consists of pairs $\langle z_{\bar{s}_{i'j}}, \bar{s}_{i'j} \rangle$ where $z_{\bar{s}_{i'j}}$ is the iteration number where $\bar{s}_{i'j}$ was added to $\Gamma_{i'}$.

5. Check if the pair $\langle z_{\bar{s}_{i'j}}, \bar{s}_{i'j} \rangle$ has been in the previous overbid list for longer than a communication loop to the agent being planned for $(\mathcal{D}_{i \to i'} + \mathcal{D}_{i' \to i})$(Line 6), and if the predicted bid has been in local bid space too long remove i' from $\mathcal{N}_i$ (Line 7) and continue checking other agents (Line 8).

6. If overbid is new, construct a pair of $iterationNumber$ and bid $\bar{s}_{i'j}$ and add to $\Gamma_{i'}$ (Line 11).

7. Return updated neighborhood $\mathcal{N}_i$ (Line 14).

7.2.3 Task Assignment Algorithm (TAA)

This implementation of HIPC can utilize any centralized task allocation algorithms with a few required features (Line 9 from Alg. 8).

- Bids on separate tasks must follow a bundle structure. Specifically, this means that bids are created in an ordered way, where the value of each bid is based on acyclic dependencies. In CBBA, bundles are constructed as ordered lists, where bids at later places in the list are dependent on the assignment of every previous task in the list. This can be implemented with a generalized score function by value of the most recent bid must be the incremental value of adding it to the overall collection of personal tasks.

- The value of tasks shared with other agents is monotonically decreasing with respect to their dependency structure. (i.e. if a bid $\mathbf{s}_1$ is dependent on a bid $\mathbf{s}_2$, then $c_1 \leq c_2$.

This can also accommodate arbitrary score functions by using a process called Bid Warping described in [4].

- Agents cannot remove any bids that they have already committed too inside of the TAA, they can only bid on unassigned tasks, or outbid a current winner of a task.

The implementation of HIPC in this work uses a centralized implementation of CBBA [40], but this is not required and may not be desired for some objective functions.

7.3 Convergence and Performance Insights

The key ideas of the proofs will be provided below with their main result. The full convergence result is only a slight modification of the result published in [7].

Theorem 5. *HIPC converges in at most $2N_t(N_a+1)\mathcal{D}$ iterations, where N_t is the number of available tasks, and N_a is the number of agents in the team.*

Proof. Roughly, each agent can only mis-predict the next task to be *locked in*, (see [7] for a precise definition of this) for $2\mathcal{D}$ iterations. This mis-prediction can cycle through every agent for a maximum of $2N_a\mathcal{D}$ number of iterations. At this point, the next task to be *locked in* will be bid on by its winning agent and this will take $2\mathcal{D}$ iterations to propagate for a total of $2(N_a+1)\mathcal{D}$ iterations to lock in each task. However, every agent now has reduced the cardinality of their neighborhood set $\mathcal{N}$ by 1. The full allocation will then be realized in at most $2N_t(N_a+1)\mathcal{D}$ iterations. □

This bound will not be tight unless $N_t = 1$ and would require an incredibly malicious set of circumstances to realize because every single prediction that each agent makes will need to give the worst possible answer (in a convergence sense). If the SA is really this bad, each neighborhood will end up containing only the agents' self and all subsequent replanning iterations will be purely plan consensus. Additionally if $N_t \gg N_a$ (which is normally true) the convergence bound becomes $\approx 2N_t\mathcal{D}$, the perfect SA worst case bound.

Theorem 6. *HIPC with imperfect situational awareness will converge to the same solution as a centralized TAA*

Proof. This result follows closely with the convergence proof. The winning bids that a centralized TAA algorithm would return are sequentially *locked in* (again see [7] for a precise definition of this) in order of largest score using exactly the same logic as the convergence proof. □

7.4 HIPC Experimental Results

To validate the claimed results, two Monte Carlo experiments were run to demonstrate expected performance. The environments were created in a room resembling the physical flight volume at the Aerospace Controls Lab as can be seen in Fig. 12. In the each of the Monte Carlo experiments run forth is work, both agents and tasks were randomly placed in the room according to a uniform distribution over the open space.

The first Monte Carlo experiment was run where situational awareness was perfectly known for a subset of the fleet. The scenario run in Fig. 13 was with 5 agents and a varying

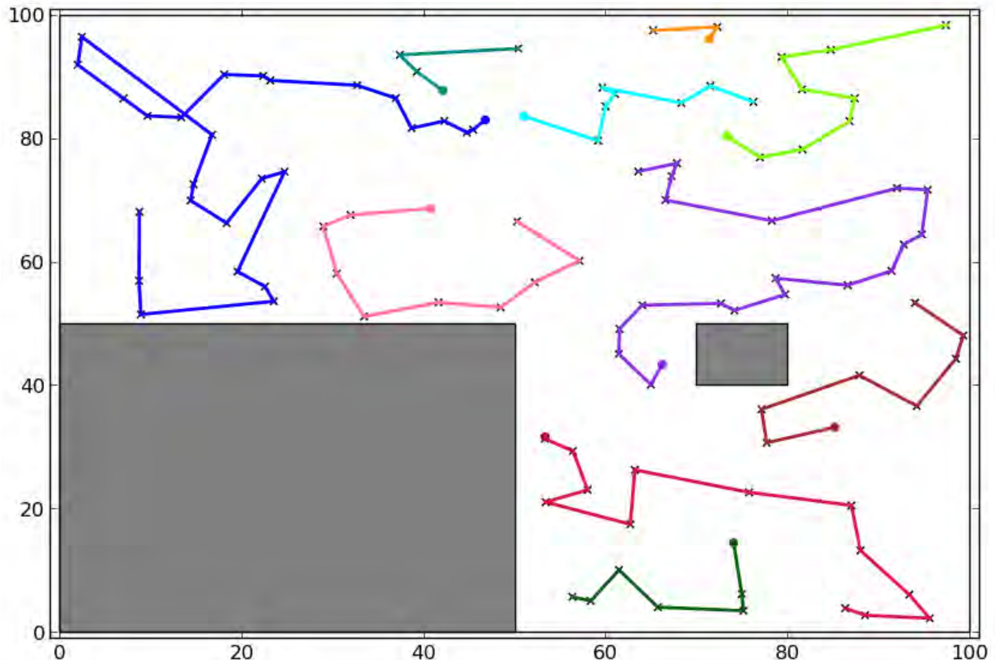


Figure 12: Figure shows an example of what the planning environment and solution would look like for a random example with 10 agents and 100 tasks. The small colored circles in the figure represent the agents initial locations, the x's are the task locations and the colored line segments represent the winning agent's estimated trajectory to service its winning bids.

number of tasks for 100 Monte Carlo iterations where the number of iterations are averaged over all of the runs. The parameter "HIPC size" refers to the number of other agents that each agent has perfect situational awareness over (i.e. HIPC size: $(|\bar{\mathcal{N}_i}| - 1), \forall i$). In this figure a dramatic reduction of convergence iterations can be seen as the HIPC size increases. The blue line in this figure, corresponding to HIPC size: 0 is the result that pure plan consensus would give (exactly the CBBA solution [40]). The magenta line in the figure, corresponding to HIPC size: 4 would be the global implicit coordination solution. All the space between these two lines consist of hybrid solutions. In the experiment the HIPC size was kept consistent for all agents, but recall this does not need to be true in practice. The HIPC size can even be asymmetric between agents planning for each other.

The second Monte Carlo experiment highlights the main contribution of hybrid approach. The results of this experiment are shown in Figure 14. This experiment was run on 7 agents with 45 tasks for 300 Monte Carlo iterations where the number of iterations is averaged over all runs. Again in this graph HIPC size refers to the number of other agents each agent is planning for $(|\bar{\mathcal{N}_i}| - 1)$ except in this example situational awareness is imperfect. The x-axis in this figure is called "Starting Location Error". The way that this environment was constructed is at each true location for a given task, a box was centered at this true task location. The imperfect locations were constructed by sampling a 2 dimensional uniform distribution where the edge lengths of the sample regions were "Starting Location Error" times the maximum dimensions of the arena. If multiple agents i were predicting the same agent i', each agent i had a unique sample for the expected location of agent i'. As expected, when the information is more consistent, planning for more agents reduces the number of

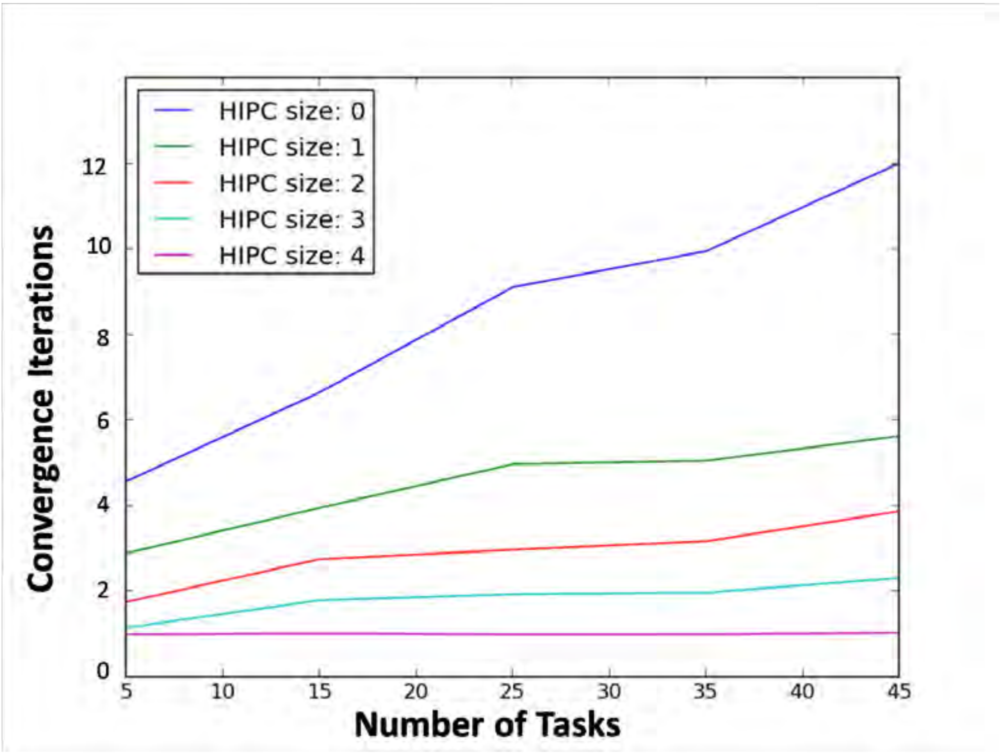


Figure 13: Figure shows the reduction in convergence iterations introduced by when perfect situational awareness is provided to HIPC. This example is a 5 agent scenario run on a varying number of tasks. HIPC size refers to the number of other agents each agent knowingly has perfect SA over (corresponding to $(|\bar{\mathcal{N}}_i| - 1)$.

convergence iterations dramatically.

An interesting case to consider is when the HIPC size $1 - 6$ lines cross above the (HIPC size: 0) line. The crossing point for each respective line corresponds to when the error in information is bad enough such that not considering any extra information would have led to faster convergence. This increase in average convergence time is a result of cases where the incorrect SA is leading to bad predictions and the agents needed to *learn* to stop planning for those agents. This learning takes more time than the savings due to predictive capability of other parts of the network. In general, missions may have a combination of perfect and imperfect situational awareness and the algorithm would handle this naturally. The convergence speed would be between the extreme cases. Also worth noting here is that for identical problem statements, all “HIPC sizes” and “Starting Location Errors” returned the same allocation with the only difference being the time to convergence. The fact that the solutions converged to the same score was part of the design of the algorithm and is a positive feature of this approach. Worth note is that for both experiments a global consistency assumption planner [36, 37] could be used, and the number of iterations for these planners would be the number of tasks assigned. This would lead to slower algorithmic convergence in each of the two experiments considered in this section

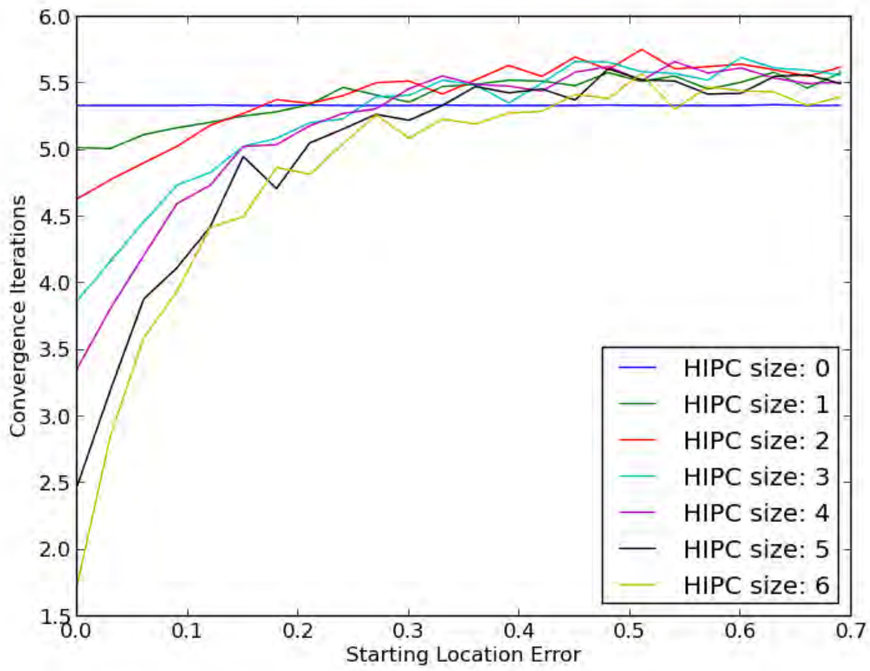


Figure 14: Figure shows the how the convergence rate of HIPC decays when situational awareness is degraded. The example used in this figure was 7 agents competing over 45 tasks. The starting location error refers to a percentage of the entire arena over which the initial location of agents in $\mathcal{N}_i$ can be wrong. HIPC size refers to the number of other agents, each agent has partial situational awareness over (corresponding to $|\mathcal{N}_i| - 1$).

7.5 Real Time Hardware Demonstrations

A video of a real time HIPC hardware demonstration (Figure 15) was produced as part of the presentation at the International Symposium on Distributed Autonomous Robotic Systems in South Korea this November. This real time experiment included 6 decentralized hardware agents with changing network connections moving about an arena servicing tasks that were arriving randomly. The right side of Figure 15 shows a visualizer of the planner state at a snapshot in time. The 2 subfigures in the left side of Figure 15 are 2 angles of live video of the demonstration that include fleet wide planner state projected down on to the ground in real time. In this screen shot, the network happens to be strongly connected (this was not true for the entire demonstration) with connections shown in purple. When an agent pair has an orange line connecting them this symbolizes that they have some noisy measurements of their partner's position and therefor they can attempt to utilize the prediction capabilities in HIPC. The dark colored X's in the video show tasks that agents plan on servicing, and the lightly colored X's correspond to tasks that have already been serviced. The dotted lines that are the same color as the X's represent the agent's predicted path while servicing the tasks.

This experiment was done in a lab setting with the network connectivity being simulated. We have been recently working on upgrading this scenario to use actual communication

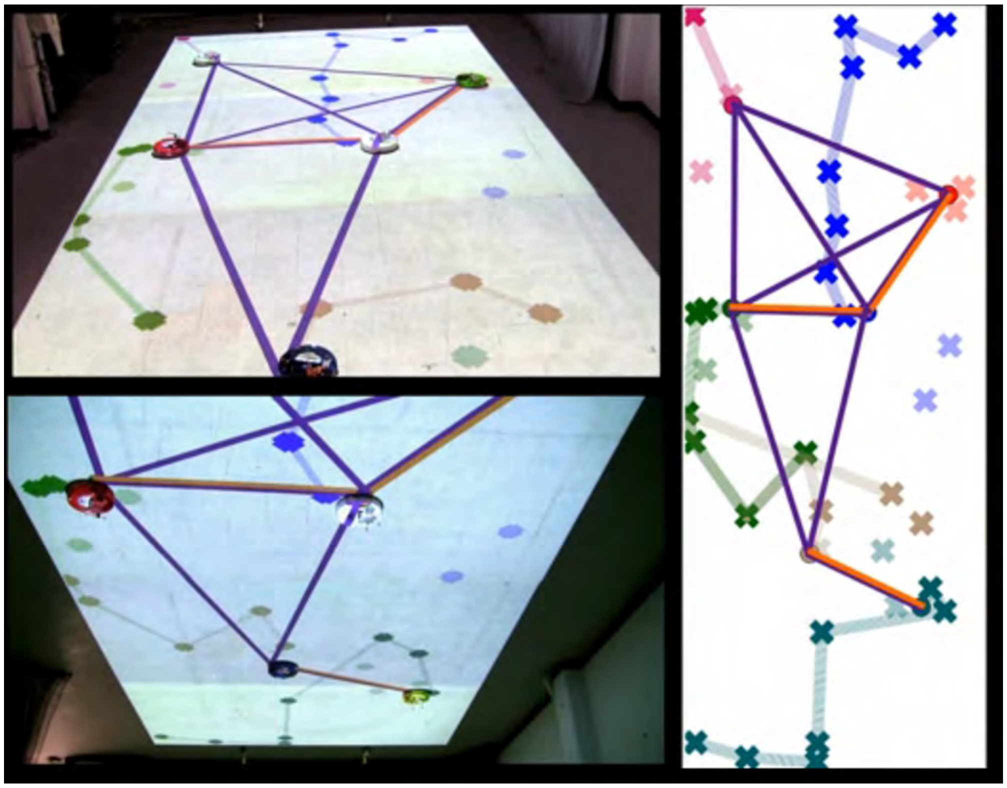


Figure 15: Figure shows a screen shot of a video of the real time hardware demonstration of HIPC presented at DARS

hardware in the loop. This work is not yet complete but the hardware platform is finalized. To do all of the onboard computations we are using Raspberry Pi B+'s (this hardware as shown in Fig. 16.)

These computers are used to do all of the computation for planning as well as handling all of the message passing. The physical communication is done using Digimesh Xbees shown attached to the Raspberry Pi's in Fig. 17. The power settings on these Xbee's are turned down all the way in order to simulate actual communication limitations. With these settings we are able to construct non-trivial networks indoors on physical hardware, that actually simulate the desired communication contested environments. Fig. 18, shows an example of this real communication environment. This shows 11 mobile agents spread throughout the 6th and 7th floors of our building. Directed lines connect nodes in the figure if there is at least a 70% chance of messages being delivered, where the red part of the line indicates the receiver of the directional link. As can be seen in the figure, the communication network is heterogeneous, disconnected, and non-symmetric. Environments like these will be used to demonstrate the downside of using GICA algorithms, and how LICA algorithms can significantly improve planning performance.

Acknowledgment/Disclaimer: This work was sponsored (in part) by the AFOSR and USAF 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 Air Force Office of Scientific Research or the U.S. Government.

Figure 16: Raspberry Pi hardware

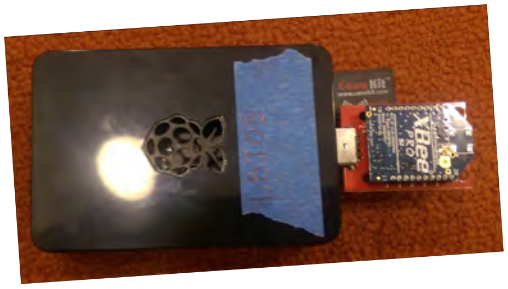


Figure 17: Raspberry Pi hardware with case and DigiMesh Xbee

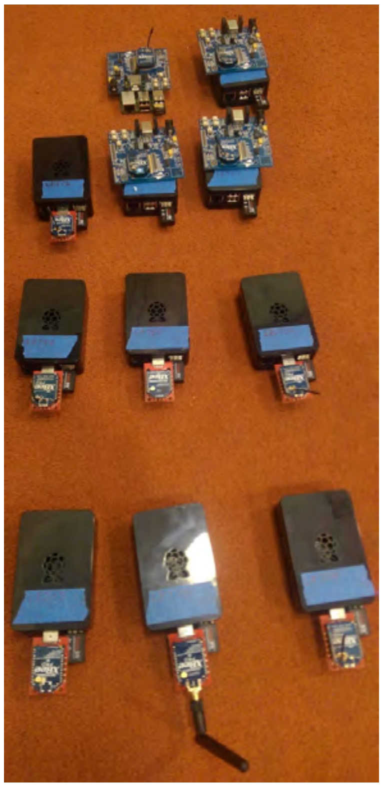


Figure 18: Team of 11 Raspberry Pi powered agents with Digimesh Xbee communication hardware.

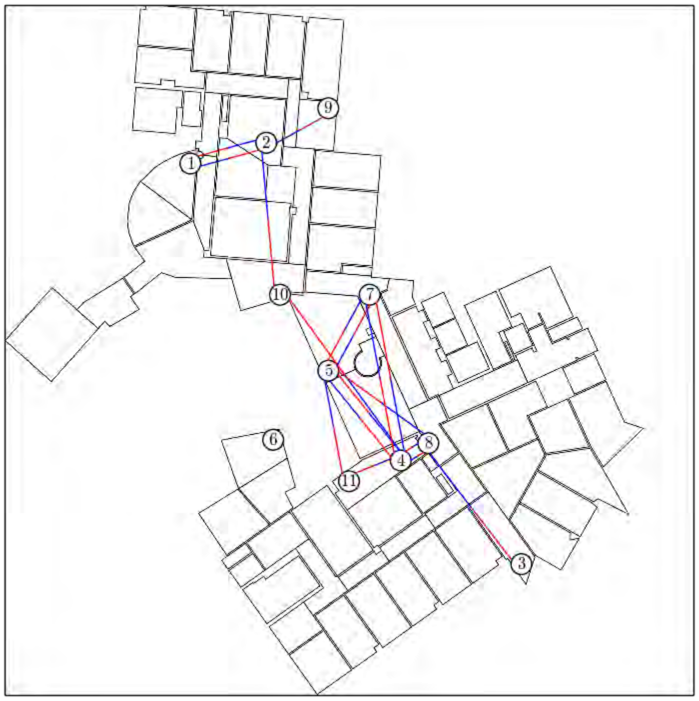


Figure 19: Raspberry Pi network in real time using DigiMesh Xbees – an example of the real communication environment using the Raspberry Pi's and DigiMesh Xbees. Shown are 11 mobile agents spread throughout the 6th and 7th floors of our building. Directed lines connect agents if there is at least a 70% chance of messages being delivered, where the red part of the line indicates the receiver of the directional link.

Personnel: Prof. How; Graduate student: Johnson

Relevant publications: See [1–10, 12–15, 47]

Honors/Awards: Professor How was: (1) Recipient of AIAA Intelligent Systems best paper presented at the 2013 AIAA Infotech@Aerospace conference (awarded 2014); (2) A recipient of the AIAA Best Paper Award from 2012 *Guidance Navigation and Control Conference* (awarded Aug 2013); (3) A recipient of the AIAA Best Paper Award from the 2011 Guidance Navigation and Control Conf. (awarded Aug 2012); (4) Awarded the prize for the "Best Applications paper published in Automatica over the past three years" in 2011.

AFRL Points of Contact: Dr. Fariba Fahroo, AFOSR/RTA.

Exchanges:

- Transitioned precursor to HIPC (CBBA with bootstrapping) to O. Toupet at Aurora (2012)
- Early collaborations with D. Casbeer and E. Garcia at AFRL for research directions
- HIPC transferred to B. Bittler at Raytheon (2014)
- HIPC transferred to Cynara Wu at STR Research (2015)
- HIPC transferred to Chris Moss at BAE (2015)

References

[1] Sameera S. Ponda, Luke B. Johnson, and Jonathan P. How. Risk allocation strategies for distributed chance-constrained task allocation. In *American Control Conference (ACC)*, June 2013.

[2] Trevor Campbell, Luke Johnson, and Jonathan P. How. Multiagent allocation of markov decision process tasks. In *American Control Conference (ACC)*, 2013.

[3] S. S. Ponda, L. B. Johnson, A. N. Kopeikin, H. Choi, and J. P. How. Distributed planning strategies to ensure network connectivity for dynamic heterogeneous teams. *IEEE Journal on Selected Areas in Communications*, 30(5):861 –869, June 2012.

[4] Luke B. Johnson, Han-Lim Choi, Sameera S. Ponda, and Jonathan P. How. Allowing non-submodular score functions in distributed task allocation. In *IEEE Conference on Decision and Control (CDC)*, Dec 2012.

[5] Luke Johnson, Sameera Ponda, Han-Lim Choi, and Jonathan P. How. Decentralized task allocation using local information consistency assumptions. *International Journal of Robotics Research*, 2015 (submitted).

[6] Luke Johnson, Han-Lim Choi, and Jonathan P. How. Hybrid information and plan consensus in distributed task allocation. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2013.

[7] Luke Johnson, Han-Lim Choi, and Jonathan P. How. Convergence analysis of the hybrid information and plan consensus algorithm. In *American Control Conference (ACC)*, Portland, OR, June 2014.

[8] Luke Johnson, Han-Lim Choi, and Jonathan P. How. The hybrid information and plan consensus algorithm with imperfect situational awareness. In *International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Daejeon, Korea, November 2014.

[9] Luke Johnson, Han-Lim Choi, and Jonathan P. How. The role of information assumptions in decentralized task allocation. *IEEE Control Systems Magazine*, 2015 (submitted).

[10] Sameera S. Ponda, Luke B. Johnson, and Jonathan P. How. Distributed chance-constrained task allocation for autonomous multi-agent teams. In *American Control Conference (ACC)*, June 2012.

[11] Sameera S. Ponda. *Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, September 2012.

[12] Andrew N. Kopeikin, Sameera S. Ponda, Luke B. Johnson, and Jonathan P. How. Multi-UAV Network Control through Dynamic Task Allocation: Ensuring Data-Rate and Bit-Error-Rate Support. In *Wi-UAV 2012, 3rd International Workshop on Wireless Networking and Control for Unmanned Autonomous Vehicles at the IEEE GlobeComm Conference*, Dec 2012.

[13] Andrew N. Kopeikin, Sameera S. Ponda, Luke B. Johnson, Olivier Toupet, and Jonathan P. How. Real-Time Dynamic Planning to Maintain Network Connectivity in a Team of Heterogeneous Unmanned Systems. In *Wi-UAV 2011, 2nd International Workshop on Wireless Networking for Unmanned Autonomous Vehicles at the IEEE GlobeComm Conference*, Dec 2011.

[14] Luke Johnson. Decentralized Task Allocation for Dynamic Environments. Master's thesis, Massachusetts Institute of Technology, January 2012.

[15] Sameera S. Ponda, Luke B. Johnson, Alborz Geramifard, and Jonathan P. How. *Handbook of Unmanned Aerial Vehicles*, chapter Cooperative Mission Planning for Multi-UAV Teams. Springer, 2012.

[16] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, 2004.

[17] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[18] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, MA, 1997.

[19] T. M. McLain and R. W. Beard. Coordination variables, coordination functions, and cooperative timing missions. *AIAA Journal on Guidance, Control, and Dynamics*, 28(1):150–161, 2005.

[20] D. A. Castanon and C. Wu. Distributed algorithms for dynamic reassignment. In *IEEE Conference on Decision and Control (CDC)*, volume 1, pages 13–18, 9-12 Dec. 2003.

[21] J. Curtis and R. Murphey. Simultaneous area search and task assignment for a team of cooperative agents. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2003 (AIAA-2003-5584).

[22] Wei Ren, R. W. Beard, and D. B. Kingston. Multi-agent Kalman consensus with relative uncertainty. In *American Control Conference (ACC)*, volume 3, pages 1865–1870, 8-10 June 2005.

[23] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology

and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[24] M. Alighanbari and J. P. How. An Unbiased Kalman Consensus Algorithm. In *American Control Conference (ACC)*, pages 3519–3524, Minneapolis, MN, 14-16 June 2006.

[25] C. C. Moallemi and B. V. Roy. Consensus propagation. *IEEE Transactions on Information Theory*, 52(11):4753–4766, 2006.

[26] A. Olshevsky and J. N. Tsitsiklis. Convergence speed in distributed consensus and averaging. In *IEEE Conference on Decision and Control (CDC)*, pages 3387–3392, 2006.

[27] Wei Ren, R. W. Beard, and E. M. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, April 2007.

[28] Y. Hatano and M. Mesbahi. Agreement over random networks. In *43rd IEEE Conference on Decision and Control*, 2004.

[29] C. W. Wu. Synchronization and convergence of linear dynamics in random directed networks. *IEEE Transactions on Automatic Control*, 51(7):1207–1210, 2006.

[30] A. Tahbaz-Salehi and A. Jadbabaie. On consensus over random networks. In *44th Annual Allerton Conference*, 2006.

[31] S. Sariel and T. Balch. Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In *AIAA Workshop on Integrating Planning Into Scheduling*, 2005.

[32] A. Ahmed, A. Patel, T. Brown, M. Ham, M. Jang, and G. Agha. Task assignment for a physical agent team via a dynamic forward/reverse auction mechanism. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2005.

[33] M. L. Atkinson. Results analysis of using free market auctions to distribute control of UAVs. In *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, 2004.

[34] T. Lemaire, R. Alami, and S. Lacroix. A distributed task allocation scheme in multi-UAV context. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3622–3627, 2004.

[35] W.E. Walsh and M.P. Wellman. A market protocol for decentralized task allocation. In *Proceedings of International Conference on Multi Agent Systems*, 1998.

[36] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *Proceedings of Robotics: Science and Systems*, 2005.

[37] Patrick Amstutz, Nikolaus Correll, and Alcherio Martinoli. Distributed boundary coverage with a team of networked miniature robots using a robust market-based algorithm. *Ann. Math. Artif. Intell.*, 52(2-4):307–333, 2008.

[38] M. Hoeing, P. Dasgupta, P. Petrov, and S. O'Hara. Auction-based multi-robot task allocation in comstar. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 2007.

[39] P. B. Sujit and R. Beard. Distributed sequential auctions for multiple UAV task allocation. In *Proceedings of the American Control Conference*, 2007.

[40] H.-L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, August 2009.

[41] J Marden and A Wierman. Overcoming the limitations of utility design for multiagent systems. *IEEE Transactions on Automatic Control*, 58(6):1402–1415, June 2013.

[42] M. G. Lagoudakis, M. Berhaultt, S. Koenigt, P Keskinocak, and A. J. Kleywegt. Simple auctions with performance guarantees for multi-robot task allocation. In *Proceedings of the IEEE/RSI International Conference on lntelllgent Robots and Systems*, 2004.

[43] D. Bertsimas and R. Weismantel. *Optimization over integers*. Dynamic Ideas Belmont, MA, 2005.

[44] S. Fugishige. *Submodular Functions and Optimization*. Annals of Discrete Mathematics, 2nd edition edition, 2005.

[45] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *The Fifth International Conference on Information Processing in Sensor Networks*, pages 2 –10, 0-0 2006.

[46] Sameera S. Ponda, Joshua Redding, Han-Lim Choi, Jonathan P. How, Matt A. Vavrina, and John Vian. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conference (ACC)*, Baltimore, MD, July 2010.

[47] A. K. Whitten, H.-L. Choi, L. Johnson, and J. P. How. Decentralized task allocation with coupled constraints in complex missions. In *American Control Conference (ACC)*, pages 1642 – 1649, June

2011.

[48] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 2005.

[49] T. Shima, S. J. Rasmussen, and P. Chandler. UAV team decision and control using efficient collaborative estimation. In *American Control Conference (ACC)*, volume 6, pages 4107–4112, 8-10 June 2005.

1.

1. Report Type

Final Report

Primary Contact E-mail
Contact email if there is a problem with the report.

jhow@mit.edu

Primary Contact Phone Number
Contact phone number if there is a problem with the report

6172533267

Organization / Institution name

MIT

Grant/Contract Title
The full title of the funded effort.

Distributed Hybrid Information and Plan Consensus HIPC

Grant/Contract Number
AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-11-1-0134

Principal Investigator Name
The full name of the principal investigator on the grant or contract.

Jonathan How

Program Manager
The AFOSR Program Manager currently assigned to the award

Fariba Fahroo

Reporting Period Start Date

06/01/2014

Reporting Period End Date

06/30/2015

Abstract

The research objective of this project was to increase the capabilities of decentralized task allocation algorithms. Progress has been made on several different fronts, including: a)~chance-constrained task allocation, b) task allocation with tasks defined as Markov Decision Processes, c) guaranteeing network connectivity during mission execution, d) allowing the use of non-submodular score functions during the decentralized allocation, and e) decreasing the convergence time by utilizing all of the information available in the network, f) hardware results that demonstrate the difficulty of planning in communication contested environments and the utility of using Local information consistency algorithms (LICA), and g) a tutorial on the basics of decentralized task allocation for a general audience is currently in revision for Control Systems Magazine.

Our work developed the Hybrid Information and Plan Consensus Algorithm (HIPC), which uses implicit coordination to plan for a subset of the team on-board each agent, then uses plan consensus to fix conflicts in the assignment constraints that may arise. By combining the ideas of local plan consensus and implicit coordination the algorithm empirically reduced the convergence time and number of messages required for distributed task allocation problems. Recent work rigorously proves convergence and provides a worst case convergence bound that is no slower than bid warped Consensus-Based Bundle Algorithm (CBBA),

requiring 2 times the number of assigned tasks times the network diameter (2 N_t D) iterations. Further work expanded this to imperfect situational awareness and a real time hardware demonstration was conducted showing the validity of this approach. Altogether, these results have significantly improved the state of the art capabilities of decentralized task allocation and work continues to refine these approaches.

Distribution Statement

This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement

If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF The maximum file size for an SF298 is 50MB.

AFD-070820-035.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The maximum file size for the Report Document is 50MB.

report.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

Luke Johnson, Sameera Ponda, Han-Lim Choi, and Jonathan P. How. Decentralized task allocation using local information consistency assumptions. International Journal of Robotics Research, 2015 (submitted).

Luke Johnson, Han-Lim Choi, and Jonathan P. How. The role of information assumptions in decen- tralized task allocation. IEEE Control Systems Magazine, 2015 (submitted).

Changes in research objectives (if any):

N/A

Change in AFOSR Program Manager, if any:

N/A

Extensions granted or milestones slipped, if any:

N/A

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Sep 14, 2015 14:59:39 Success: Email Sent to: jhow@mit.edu