View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by DSpace@MIT

Allowing Non-Submodular Score Functions in Distributed Task Allocation

Luke Johnson, Han-Lim Choi, Sameera Ponda and Jonathan P. How

***Abstract*— Submodularity is a powerful property that can be exploited for provable performance and convergence guarantees in distributed task allocation algorithms. However, some mission scenarios cannot easily be approximated as submodular a priori. This paper introduces an algorithmic extension for distributed multi-agent multi-task assignment algorithms that enables them to converge using non-submodular score functions. These enhancements utilize 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. It is proven that external to each agent, it seems as if a submodular score function is being used (even though this function is never explicitly created), and thus all convergence and performance guarantees hold with respect to this apparent submodular score function. The results of this effort show significant improvements over hand-tuned heuristic approaches that approximate the true non-submodular score functions.**

I. Introduction

Distributed and decentralized task allocation algorithms are used in environments where communications may be unreliable due to high latency, high cost, or simply being unavailable. As a result, distributed planning methods which eliminate the need for a central server have been explored [11], [6], [17]. Many of these methods often assume perfect communication links with infinite bandwidth in order to ensure that agents have the same situational awareness before planning. In the presence of inconsistencies in situational awareness, these distributed tasking algorithms can be augmented with consensus algorithms [15], [13], [12], [2], [14] to converge on a consistent state before performing the task allocation. Although consensus algorithms guarantee convergence on information, they may take a significant amount of time and often require transmitting large amounts of data [3]. Other popular task allocation methods involve using distributed auction algorithms [16], [1], which have been shown to efficiently produce sub-optimal solutions. Two recent approaches are a distributed algorithm called the consensus based bundle algorithm (CBBA) [5] and a decentralized asynchronous version called the asynchronous consensus based bundle algorithm (ACBBA) [9]. These task allocation algorithms use a greedy auction to produce multi-task assignments for each agent. These algorithms can produce provably good solutions with respect to the optimal if the score function obeys a property called submodularity. The importance of submodularity was identified in Section IV.C.1 [5] for CBBA, and the form it takes in these algorithms is called Diminishing Marginal Gains (DMG). This DMG property is required for both the convergence and performance guarantees made for CBBA, and score functions that violate this property cannot be used directly within the algorithmic framework but must be modified using heuristics that satisfy submodularity. While DMG ensures convergence of the algorithm, it can be a limiting constraint for many problems of interest in which the true score function is not submodular. Previous work identified modifications to the score that employ heuristics to ensure that DMG is satisfied [5], but these heuristics typically lead to poor performance and are not usually intuitive to design. The main contribution of this paper is to provide new algorithmic modifications to the CBBA algorithm itself that enable the use of non-DMG score functions, and thus lead to plans that more closely align with mission designer intentions.

A. Problem Formulation

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

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

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 $\mathbf{H} = \begin{bmatrix}\mathbf{h}_1 \dots \mathbf{h}_{N_c}\end{bmatrix}^T$, with $\mathbf{d} = \begin{bmatrix}d_1 \dots d_{N_c}\end{bmatrix}^T$, 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 generalized problem formulation can accommodate several different design objectives and constraints commonly 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 very difficult to solve (NP-hard) due to the inherent system inter-dependencies [4].

The algorithms used in this paper 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 greedy assignments for themselves, and then iterate with others using a consensus algorithm to produce a final fleet-wide assignment. The details of this process will be explored throughout the rest of this paper.

II. Background

The submodularity condition is based on a well defined and extensively studied mathematical construct [7]. As applied to the distributed greedy allocation problem it can be defined as follows: the marginal score function $F(s, \mathcal{A})$ for adding a task s to an existing task environment $\mathcal{A}$, must satisfy the following

$$F(s, \mathcal{A}') \geq F(s, \mathcal{A}) \quad \forall \mathcal{A}' \; s.t. \; \mathcal{A}' \subset \mathcal{A} \tag{2}$$

Equation (2) roughly means that a particular task cannot increase in value because of the presence of other assignments. Although many score functions typically used in task allocation satisfy this submodularity condition (for example the information theory community [10]), many also do not.

It is simple to demonstrate that the distributed greedy multi-assignment problem may fail to converge with a non-submodular score function, even with as few as 2 tasks and 2 agents. Consider the following example, where notation for an agent's task group is (task ID, task score), and the sequential order added moves from left to right. The structure of this simple algorithm is that each agent produces bids on a set of desired tasks, then shares these with the other agents. This process repeats until no agent has incentive to deviate from their current 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. In the above example, ϵ is defined as some value, $0 < \epsilon < T$.

Example 1: **Allocations with a submodular score function**

`Iteration 1`
- Agent 1: $\{(1, T), (2, T - \epsilon)\}$
- Agent 2: $\{(2, T), (1, T - \epsilon)\}$

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

In `Iteration 1`, each agent bids on both tasks 1 and 2, but the bid value for the second task placed by both agents is ϵ less than the bid for the first task. This result is consistent with submodularity since the score has not increased because of the presence 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 greedily maximizes the global score.

The following example highlights how convergence breaks when non-submodular score functions are used. In this example, ϵ is defined to be some value, $0 < \epsilon < \infty$. In `Iteration 1` both agents again bid on both tasks, however, this time the score on the second task has increased because of the presence of the first task. This explicitly violates the submodularity condition as presented in Equation (2). Like before, between `Iteration 1` and `Iteration 2` the agents share their bids with each other. In `Iteration 2` both agents have been outbid on their first task and thus their bids on their second tasks are invalidated. For simplicity assume that without the presence of the first task, neither agent can even place a positive bid on their second task[1] and thus for this iteration neither agent places bids. `Iteration 3` then exactly repeats `Iteration 1` and the cycle will repeat forever.

Example 2: **Allocations with a non-submodular score function**

`Iteration 1`
- Agent 1: $\{(1, T), (2, T + \epsilon)\}$
- Agent 2: $\{(2, T), (1, T + \epsilon)\}$

`Iteration 2`
- Agent 1: $\{\}$
- Agent 2: $\{\}$

`Iteration 3`
- Agent 1: $\{(1, T), (2, T + \epsilon)\}$
- Agent 2: $\{(2, T), (1, T + \epsilon)\}$

One may wonder if there is a simple fix to detecting these cycles. The short answer is no, because these cycles can take the form of an arbitrary number of tasks being traded between an arbitrary number of agents during the consensus process. Even worse, is that breaking out of one cycle does not guarantee that the agents will enter another cycle at a later stage in the convergence process. In practice, any non-submodular score functions very often lead to fairly malicious cycling behavior. The following example will attempt to highlight that these conditions are not exotic and in fact occur in many fairly simple environments.

Example 3 Consider the potential task environment illustrated in Figure 1 involving one agent (blue) and two tasks (red). For the purpose of this example assume that $d_{a2} > d_{a1} >> d_{12}$. An intuitive score function $F_{ij}(\mathbf{x}, \boldsymbol{\tau})$ for this

[1]This assumption is required for a 2 iteration cycle. Other assumptions can still produce cycles but these would be longer and involve more complex bid histories.

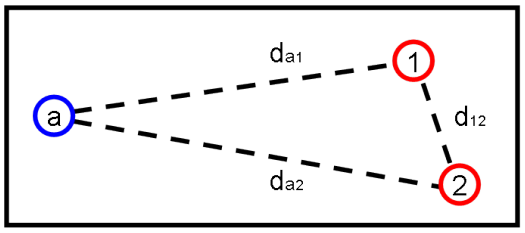


Fig. 1: A potential task environment where **a** represents the agent, and **1** and **2** are task locations. The notation d_{uv} is the distance measured from location u to location v.

environment is defined as follows:

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

where 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 traveled by inserting task j into the current assignment $\mathbf{x}$. If a multi-assignment greedy algorithm were run in this environment, it would first assign task 1 because $R - f_i d_{a1} > R - f_i d_{a2}$. When the greedy 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 this can easily lead to cycling. A typical strategy utilized to solve this problem is to *submodularize* the score function using a heuristic. For example, the score function in Equation (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 d_{aj} represents the distance measured from the agent's initial location to task j. With this score function the first bid will again be on task 1, $R - f_i d_{a1} > R - f_i d_{a2}$, and the second bid will be on task 2, but this time the bid will have the score $R - f_i d_{a2}$. This preserves the submodularity result since the score on task 2 did not increase because of the previous assignment of task 1. However, this score function is unable to capture the fact that, since task 1 is being serviced, task 2 should seem much more favorable. The purpose of the approach presented in this paper is to enable the algorithms to use score functions that do capture these non-submodular effects without having to sacrifice convergence guarantees.

A. Allowing non-submodular score functions

The convergence failures highlighted above in Example 2 are a direct result of multiple tasks being assigned simultaneously. [8] identifies a class of algorithms that can utilize distributed computation and can converge using non-submodular score functions. However, the thesis argues that the algorithms mentioned are not good distributed and decentralized implementations because: 1) they require that every iteration of the algorithm be synchronized across the entire fleet (a very hard constraint to enforce in real systems), and 2) require many more iterations to converge in practice than algorithms that assign multiple tasks simultaneously. For these reasons, this paper focuses on augmenting algorithms that can assign multiple tasks *simultaneously* with the ability to converge using non-submodular score functions. The approach exploits the algorithmic structure associated with the sharing of information in distributed and decentralized algorithms.

1) Since each distributed agent is isolated from all other agents, the only task allocation interaction that each agent makes with each other is through distributed messages with bid information.
2) Given insight 1, it is not really the score function that needs to be submodular. It is the bid values that need to *appear* as if they were created using some submodular score function to ensure that the fleet will converge.

Using the insights provided above, it is possible for an agent to use a non-submodular score function to rank prospective tasks, then rely on some externally visible *warping function* to produce bids that are consistent with submodularity. If the external bids are created correctly, to all other agents participating in the task allocation process it will appear as if there were some submodular score function creating those bids. Thus algorithms that rely on submodularity (including CBBA and ACBBA) will actually converge. The main improvement comes from the fact that ranking the tasks internally in a non-submodular way allows each agent to retain some structure present in the non-submodular score function, while still producing submodular bids. The following section will outline exactly how this is achieved.

III. Solution Approach

This section focuses on the algorithmic modifications specific to CBBA, but similar techniques may be used for other multi-assignment task allocation problems.

A. Baseline CBBA

CBBA is a distributed auction algorithm that provides provably good task assignments for multi-agent, multi-task allocation problems. The algorithmic structure of CBBA is an iterative, 2 phase algorithm. These two phases are: a *bundle building* phase where each vehicle greedily generates an ordered bundle of tasks, 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: $s_{ij} = (i, j, c_{ij})$, where i represents the bidding agent's index, j represents the task's index, and c_{ij} represents the bid value 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 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.

Algorithm 1 CBBA: Bundle Building Phase

procedure BUILD BUNDLE($\mathcal{A}_i$)
  for all k such that $s_{ij_k} \in \mathcal{A}_i$ **do**
    $\mathcal{A}_i \setminus s_{ij_k}$
  end for
  set $\mathbf{b}_i = \emptyset$
  while $|\mathbf{b}_i| < L_t$ **do**
    for all $j \in \{1, \ldots, N_t\} \setminus \mathbf{b}_i$ **do**
      $c_{ij} = F_{ij}(\mathbf{b}_i)$,
      $h_{ij} = \mathbb{I}(c_{ij} > s_{i'j} \in \mathcal{A}_i),\ \forall i' \in \{1, \ldots, N_a\}$
    end for
    $j^\star = \text{argmax}_j\, c_{ij} \cdot h_{ij}$
    $s_{ij^\star} = (i, j^\star, c_{ij^\star})$
    if $c_{ij^\star} > 0$ **then**
      $\mathbf{b}_i = \mathbf{b}_i \oplus s_{ij^\star}$
    else
      break
    end if
  end while
end procedure

3) The **bid space** is an unordered set of bids, defined as $\mathcal{A} = \{s_{i_1 j_1}, \ldots, s_{i_N j_N}\}$. This set contains a globally consistent set of the current winning bids in the fleet. 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 consensus phase, but in general, the geometry of agents in the network may lead to information propagation latencies and thus non-identical bid spaces.

Bundle Building Phase For each agent i the bundle building phase is run independently.

1) At the beginning of the bundle building phase, the previous bundle is cleared and all bids that were won by agent i located in the local bid space, $\mathcal{A}_i$, are also removed. This step is required for the optimality guarantees of the algorithm[2], but in most cases, the agent will re-add each of the tasks it has just dropped from $\mathcal{A}_i$ as it is building back up the bundle.
2) For each task j available in the environment, each agent i uses its local internal score function $F_{ij}(\mathbf{b}_i) = c_{ij}$, which is a function of its current bundle, to create a score c_{ij}. These scores are then ordered from highest to lowest.
3) The ordered scores c_{ij} are compared with the winning bid information for the corresponding task j located in the agent's local bid space $\mathcal{A}_i$. The first (and thus largest) score that would outbid the current winner in the local bid space is chosen as agent i's next bid. A bid s_{ij} is created and placed at the end of the bundle, and replaces the current bid on task j in the local bid space. Steps 2 and 3 are repeated until no tasks are able to outbid $\mathcal{A}_i$ 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 values s_{ij} are communicated as bids to other agents. The main result of this paper is to separate these two values in order to enable a larger class of score functions.

[2]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.

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 [5], 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 DMG satisfying score function. This DMG condition is a subset of the submodularity condition introduced in the background section, and was recognized in the seminal description of CBBA [5]. It is defined as

$$F_{ij}(\mathbf{b}_i) \geq F_{ij}(\mathbf{b}_i \oplus_{\text{end}} s_{ik^\star}) \quad \forall k^\star \in \{1, \ldots, N_t\} \tag{5}$$

where $\mathbf{b}_i \oplus_{\text{end}} s_{ik^\star}$ refers to adding a bid $s_{ik^\star}$ to an already existing bundle $\mathbf{b}_i$. Roughly this condition means that no bids $s_{ik^\star}$ can be made that would increase c_{ij}, agent i's score for task j.

B. Expanding CBBA to Handle Non-DMG Score Functions

The approach introduced in this paper changes two fundamental aspects of placing bids. First, the ranking of task scores is allowed to use a function that need not satisfy DMG; second, the external bid values are not identical to the scores used for the internal ranking, and are created in a way such that they are guaranteed to satisfy DMG. To highlight the algorithmic changes an additional piece of notation is needed.

1) A **bid warping function**, $G_{ij}(c_{ij}, \mathbf{b}_i) = \bar{c}_{ij}$, is a function that produces an output $\bar{c}_{ij}$ based on the internal score function c_{ij} and the current bundle $\mathbf{b}_i$. This function G is defined as

$$\bar{c}_{ij} = \min(c_{ij}, \bar{c}_{ij_k}) \quad \forall k \in \{1, \ldots, |\mathbf{b}_i|\} \tag{6}$$

where $\bar{c}_{ij_k}$ is the score of the kth element in the current bundle.

C. Distributed Greedy Algorithms with Non-Submodular Score Functions

This section presents the main algorithmic modifications required for CBBA to use non-submodular score functions.

Bundle Building Phase Again, for each agent i, the bundle building phase is run independently.

Algorithm 2 CBBA: Bundle Building with Non-DMG Scores

procedure BUILD BUNDLE($\mathcal{A}_i$)
  for all k such that $\bar{s}_{ij_k} \in \mathcal{A}_i$ **do**
    $\mathcal{A}_i \setminus \bar{s}_{ij_k}$
  end for
  set $\mathbf{b}_i = \emptyset$
  while $|\mathbf{b}_i| < L_t$ **do**
    for all $j \in \{1, \ldots, N_t\} \setminus \mathbf{b}_i$ **do**
      $c_{ij} = F_{ij}(\mathbf{b}_i)$
      $\bar{c}_{ij} = G_{ij}(c_{ij}, \mathbf{b}_i)$
      $h_{ij} = \mathbb{I}(\bar{c}_{ij} > \bar{s}_{i'j} \in \mathcal{A}_i),\ \forall i' \in \{1, \ldots, N_a\}$
    end for
    $j^\star = \text{argmax}_j\, c_{ij} \cdot h_{ij}$
    $\bar{s}_{ij^\star} = (i, j^\star, \bar{c}_{ij^\star})$
    if $\bar{c}_{ij^\star} > 0$ **then**
      $\mathbf{b}_i = \mathbf{b}_i \oplus \bar{s}_{ij^\star}$
    else
      break
    end if
  end while
end procedure

1) At the beginning of the bundle building phase, the previous bundle is cleared and all bids that were won by agent i located in the local bid space, $\mathcal{A}_i$, are also removed.
2) For each task j available in the environment, each agent i uses its local internal score function $F_{ij}(\mathbf{b}_i) = c_{ij}$, which is a function of its current bundle, to create a score c_{ij}. The only 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 with identical initial conditions the function returns an identical score.
3) The score values c_{ij} are then warped using the bid warping function $G_{ij}(c_{ij}, \mathbf{b}_i) = \bar{c}_{ij}$ to create the warped bids $\bar{c}_{ij}$.
4) Each of the $\bar{c}_{ij}$ are compared with the winning bid for the corresponding task j located in the local bid space $\mathcal{A}_i$. The task j 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. A bid $\bar{s}_{ij}$ is created and placed at the end of the bundle, and also replaces the current bid on task j in the local bid space. If no tasks are able to outbid $\mathcal{A}_i$ or the maximum bundle length L_i is reached, the bundle building phase terminates; if not, Step 2 and 3 are repeated. The key insight in this algorithm is that the value c_{ij} is used to rank the tasks but the warped bid $\bar{s}_{ij}$ is what is actually shared with the other agents.

Consensus Phase There are no changes to the algorithm in this phase. There is a slight theoretical change because the algorithm is now using the warped bid to perform consensus instead of the value produced by each agent's internal score function. Because of the form in which this bid was created, the algorithm will converge as if it were using a DMG satisfying score function.

The end result of these two modifications is that, to an external agent, it seems as if there is some submodular cost function that is creating bids. In actuality, the agent is using a non-submodular cost function to make decisions on local bids, then warping these bid values when sharing them with other agents.

D. Bids Guaranteeing DMG

This section proves that every bid produced with the warping function G satisfies DMG. For this proof, the following assumption is needed: given identical initial conditions (local bid space $\mathcal{A}_i$ and bundle $\mathbf{b}_i$), the subsequent bid values are repeatable. This condition still allows for score functions that are stochastic, however, the stochastic value must be sampled deterministically. This can be done by either enforcing a common seed when simulating the stochastic score, or by saving the stochastic particles used to compute the bids.

Theorem 1: If bundles are fully reconstructed at every bundle building phase, the resulting bundles are always monotonically decreasing in score.

Proof: By induction: Base Case $|\mathbf{b}_i| = 1$. The first bid in the bundle will be $\bar{s}_{ij_1}$. By definition, a single element will be monotonically decreasing. Next assume that the bundle $\mathbf{b}_i$, where $|\mathbf{b}_i| = n$, is monotonically decreasing. According to the definition of the bid warping function, the bid values are defined as

$$\mathbf{b}_i(n+1) = \bar{s}_{ij_{n+1}} = \min(\bar{s}_{ij_n}, s_{ij_{n+1}}) \tag{7}$$

therefore $\bar{s}_{ijn+1} \leq \bar{s}_{ij_n} \implies \mathbf{b}_i(n+1) \leq \mathbf{b}_i(n)$ and the bundle $\mathbf{b}_i$ is monotonically decreasing. ■

Theorem 2: If a bundle is monotonic, $\exists$ some DMG satisfying score function $\bar{G}$ that would produce an identical bundle (tasks and scores).

Proof: By induction: Base case $|\mathbf{b}_i| = 1$. The bid located in position 1 of agent i's bundle, $\bar{s}_{ij_1}$, will always satisfy DMG as long as bids are repeatable. Assume up to a bundle length of n a DMG satisfying score function $\bar{G}$ could have produced the bundle $\mathbf{b}_i$. Given monotonicity implied by Theorem 1, $\exists \alpha_k \geq 0$ s.t.:

$$\begin{aligned} \bar{G}_{ij_k}(\mathbf{b}_i) &= \bar{G}_{ij_{n+1}}(\mathbf{b}_i) + \alpha_k, \\ |\mathbf{b}_i| &= k-1,\ \ \forall k = \{1, \ldots, n\} \end{aligned} \tag{8}$$

where $\bar{G}_{ij_k}(\mathbf{b}_i)$ is the DMG satisfying score function for the k^{th} task in agent i's bundle that is a function of the first $k-1$ elements in the bundle. $\bar{G}_{ij_{n+1}}(\mathbf{b}_i)$ is the score function for the task that will be the $(n+1)^{th}$ element of the bundle, evaluated when the bundle only contained the first $k-1$ elements. When constructing the function $\bar{G}$, it is possible to pick a set of α_k such that

$$\begin{aligned} \bar{G}_{ij_{n+1}}(\mathbf{b}_i) &= G_{ij_{n+1}}(\mathbf{b}_i^\star) = \bar{c}_{ij_{n+1}} \\ |\mathbf{b}_i^\star| = n, |\mathbf{b}_i| &= k-1,\ \ \forall k = \{1, \ldots, n+1\} \end{aligned} \tag{9}$$

where $G_{ij_{n+1}}(\mathbf{b}_i)$ is the actual bid warping score function (defined in Section III-B), which is also equal to the actual

bid placed on task j_{n+1}, $\bar{c}_{ij_{n+1}}$. Since in the relation above, $\bar{G}_{ij_{n+1}}$ is identical for all subsets of the length n bundle, $\mathbf{b}_i^\star$, this relation trivially implies that

$$\bar{G}_{ij_{n+1}}(\mathbf{b}_i) = \bar{G}_{ij_{n+1}}(\mathbf{b}_i \oplus_{\text{end}} s_{ij_k}) \quad (10)$$
$$|\mathbf{b}_i| = k-1, \;\; \forall k = \{1, \ldots, n\}$$

which satisfies DMG. ■

Theorem 3: If two score functions G and $\bar{G}$ produce identical bundles, then they produce the exact same intermediate and final bid spaces.

Proof: This result is a relatively trivial result. If each agent i is considered as a black box that, given a bid space, returns a candidate bundle, it follows that if two internal score functions produce the same result it will be impossible to tell the difference between them, and thus the fleet-wide bid spaces will progress identically. ■

Since there exists a DMG satisfying score function that would have produced the exact same intermediate and final bid spaces as those produced by the warping function G (by combining Theorems 2 and 3), all provable properties about the algorithm's performance and optimality guarantees hold w.r.t. a DMG score function $\bar{G}$.

E. Dealing with Non-Deterministic Score Functions

As was mentioned above, it is important to have repeatable score functions. This actually accommodates score functions with some stochastic behavior. If the algorithm uses the same set of particles for sampling the uncertainty, or the same algorithm seed at every evaluation of the score function, then the function will return a repeatable value. Truly stochastic function evaluations, however, do not have absolute convergence guarantees with this approach. Certain distributions may converge with high probability, but these special cases are not further explored in this paper.

F. Alternate Approach

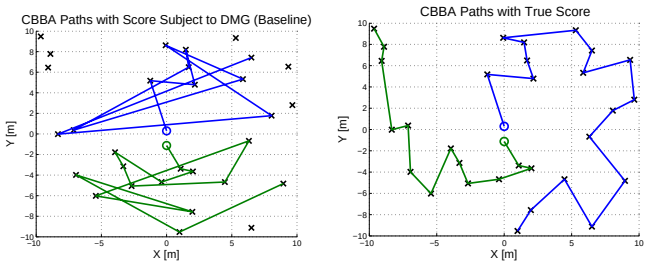
It was postulated as Lemma 4 in [5] that a trick for augmenting the score function to satisfy the DMG condition would be to ensure that the bids were monotonic in subsequent iterations:

$$\bar{c}_{ij}(t) = \min\{c_{ij}, \bar{c}_{ij}(t-1)\} \quad (11)$$

where c_{ij} is the initial score at iteration t and $\bar{c}_{ij}(t)$ 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 clearly not a desired result and a more elegant solution has been presented in this paper.

IV. Case Study: Consensus Based Bundle Algorithm

The algorithm improvements described in Section III-C were implemented within the distributed CBBA framework to enable the use of non-submodular score functions. This section presents results that compare the performance of this improved algorithm against the original CBBA baseline algorithm in 3 different domains that highlight key improvements associated with this new approach.



(a) Paths using submodular heuristic (b) Paths using true score functions

Fig. 2: 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 that satisfies DMG. Fig (b) shows the planned paths for the same scenario using CBBA augmented to handle non-submodular (true) score functions as proposed in Section III-C.

The first example considers a simple mission where agents must travel to and service tasks at different locations. The score function associated with this mission is defined as:

$$J = \sum_{i=1}^{N_a} \left(50 \left(\sum_{j=1}^{N_t} (x_{ij}) \right) - f_i d_i(\mathbf{b}_i) \right) \quad (12)$$

where a reward of 50 is obtained for each task visited, and again a fuel cost f_i is multiplied by the distance traveled $d_i(\mathbf{b}_i)$ by agent i for its assigned set of tasks $\mathbf{b}_i$. Figure 2 compares the planned paths for a 2 agent, 30 task mission, using the original baseline CBBA algorithm (left) and the new CBBA augmented to handle non-submodular score functions (right). As explained in Figure 1 of Section II, a heuristic approach to ensure submodularity within the original CBBA framework involves approximating the fuel penalty in Equation (12) by a constant value based on the initial agent position and the task location. This heuristic score function cannot explicitly capture how order matters in task execution, producing task scores that are independent of previous tasks in the agent's bundle. This results in the algorithm selecting tasks that are closest to the agent's *initial* position instead of correctly predicting the agent position based on tasks already in the agent's bundle (Figure 2(a)). On the other hand, enabling the use of non-submodular cost functions allows the algorithm to capture the benefit associated with doing tasks that are closer to those already in the agent's bundle, leading to higher scoring paths (Figure 2(b)).

Figure 3 presents Monte Carlo simulation results for a 6 agent team as a function of the number of tasks in the environment. As shown in the plot, the new CBBA approach outperforms baseline CBBA, achieving much higher mission scores. For each scenario, a centralized sequential greedy algorithm was also run for comparison. The full description of this centralized algorithm is introduced in

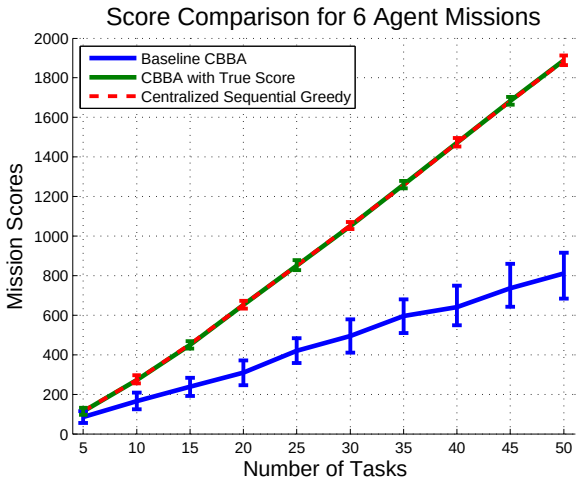


Fig. 3: Monte Carlo simulation results for a 6 agent team. Results show that the distributed CBBA approach, augmented to handle non-submodular score functions, achieves higher performance than baseline CBBA and similar scores to the centralized sequential greedy planner.

[8] but its two relevant features are: 1) it returns the same solution as CBBA for submodular score functions and 2) it retains guaranteed convergence for non-submodular score functions. The results from Figure 3 show that the distributed CBBA approach augmented to enable non-submodular score functions achieves the same performance as the centralized sequential greedy algorithm. Although these are not always guaranteed to be equal for all non-submodular score functions, for many natural problems of interest (such as the example presented above), the distributed solution tends to match the centralized one. Another important thing to note is that the non-submodular score function allows the algorithm to accurately predict the mission score, whereas the heuristic score function leads to score predictions that are usually inaccurate. In addition to increased confidence in the planner's predictions, this effect has two main benefits. First, the variability of the produced plan is lower, since the algorithm can capture the true dynamics of the mission, and the plan quality therefore is not associated with choosing a good or bad DMG satisfying heuristic. This can be seen in Figure 3 where the error bars for the baseline CBBA case are much larger than those of the planners using the true score function. The second benefit is that plan stability increases; to mitigate the effect of the heuristic approximation, one can replan as the agent location changes, however, the plans produced after each consecutive replan are likely to look quite different. Using the true non-DMG satisfying score function during the task allocation process leads to plans that are accurate initially, therefore replanning is not likely to change the solution much, and thus the task allocations are consistent across replans.

The third scenario considered involves a similar mission, where agents must service tasks at different locations, however this time there is uncertainty in the planning parameters and task rewards are time-critical. In particular, the agents have uncertain velocities and service times for tasks, although the distributions of these are known a priori (log-normal). The objective is to maximize a risk-aware robust mission score, where the probability of obtaining a score lower than predicted is below a certain threshold, ϵ. In this stochastic setting, the DMG condition is broken because the stochastic metrics implicitly couple the effects of parameter uncertainty, and therefore couple task scores. This leads to dependencies between tasks which, as illustrated in the example of Figure 1, 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 the CBBA algorithmic modifications 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 DMG violation 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 as presented in the previous case study (with non-submodularity caused by fuel penalties). Figure 4 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 CBBA approach with non-submodular scores, and a baseline deterministic CBBA with submodular scores (a submodular approximation to the true robust problem). Once again the new CBBA approach achieves similar performance to the centralized planner and clearly outperforms the baseline CBBA approach.

V. Conclusion

Submodularity is a powerful property that can be exploited for provable performance and convergence guarantees in distributed task allocation algorithms. However, some mission scenarios cannot easily be approximated as submodular a priori. This paper introduced an algorithmic extension for distributed multi-agent multi-task assignment algorithms that enabled them to converge using non-submodular score functions. 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. It was proved that external to each agent, it seems as if a submodular score function is being used (even though this function is never explicitly created), and thus all convergence and performance guarantees hold with respect to this apparent submodular score function. The results of this effort showed significant improvements over hand-tuned heuristic approaches that approximate the true non-submodular score functions.

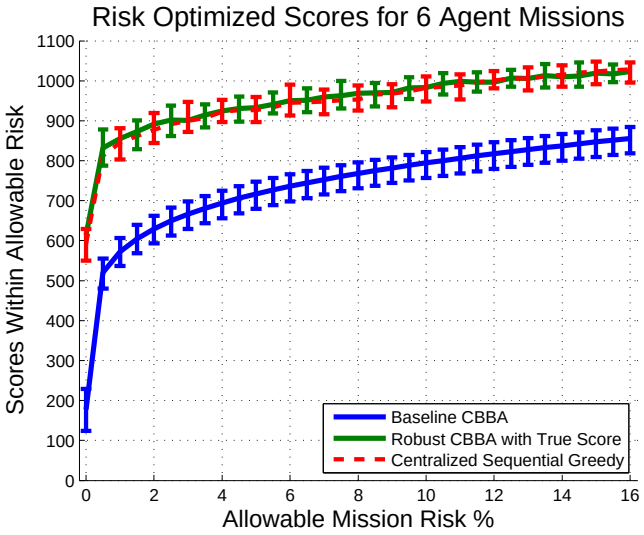


Fig. 4: Monte Carlo results for a 6 agent, 60 task, time-critical mission, with uncertain travel times and task durations. Results show risk-optimized robust mission scores for the different planning approaches.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] 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.

[2] M. Alighanbari, L. Bertuccelli, and J. How. A Robust Approach to the UAV Task Assignment Problem. In *IEEE Conference on Decision and Control (CDC)*, pages 5935–5940, 13–15 Dec. 2006.

[3] M. Alighanbari and J. How. Decentralized task assignment for unmanned aerial vehicles. In *IEEE Conference on Decision and Control (CDC)*, pages 5668–5673, 12–15 Dec. 2005.

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

[5] 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.

[6] 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).

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

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

[9] L. B. Johnson, S. Ponda, H.-L. Choi, and J. P. How. Improving the efficiency of a decentralized tasking algorithm for UAV teams with asynchronous communications. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2010 (AIAA-2010-8421).

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

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

[12] 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.

[13] W. Ren and R. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, May 2005.

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

[15] W. 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.

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

[17] 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.