2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

A Multi-Team Extension of the Consensus-Based Bundle Algorithm

Matthew Argyle, David W. Casbeer, and Randy Beard

Abstract— **The Consensus-Based Bundle Algorithm (CBBA) is incorporated into a hierarchical concept of operation. In the Team CBBA each team of unmanned vehicles plans for all agents in the team to service a set of tasks. This team planning is carried out separately using the traditional CBBA. An "outer-loop" Team CBBA strategy is presented that coordinates planning between teams of agents. The hierarchical structure of the Team CBBA gives an manageable architecture for large numbers of unmanned agents through human centered operations. This is because each (small) team would be managed by a human operator with the Team CBBA coordinating between teams.**

I. Introduction

Over the last several years, there has been significant focus in cooperative control of multi-agent systems, including task assignment problems. In a task assignment problem, a group of agents service a finite number of tasks. Solutions to task assignment problems fall under two broad categories: centralized or decentralized. Centralized methods involve all the agents sending all relevant data to one agent/ground station which then computes and returns a plan for all agents. Centralized methods typically yield better solutions than decentralized methods with less computational burden on the (non-central) agents; however, there is a single point of failure and the solution depends on successful and full communication.

From an operational standpoint, centralized methods fit easily into the human centered model, where an operator manages the team of agents. Since the operator knows the location of each agent in the team, it is logical to have planning take place centrally. One could naively extend this concept of operation to large numbers of unmanned agents, by giving multiple human operators independent control over separate teams. However, mission performance would be improved through coordination between the teams.

This paper presents a decentralized algorithm based on the Consensus-Based Bundle Algorithm (CBBA) [1], [2], [3], [4]. Inspired by the human centered model, in this paper we extend the CBBA to a hierarchical team structure by incorporating an "outer-loop" Team CBBA. In this hierarchical model, teams of unmanned agents are managed by a human operator. Planning is accomplished on each team using the traditional CBBA [1] and coordination between teams is carried out by a Team CBBA. Section II lays out the specific problem we solve. Section III discuss the multi-team extension to the CBBA, which is followed by analysis in Section IV. We then present simulations results and conclusions in Sections V and VI.

M. Argyle is an AFRL Summer Researcher and a PhD student with the Department of Electrical Eng., Brigham Young University, Provo, Utah, 84602 matt.argyle@gmail.com

D. Casbeer is with the Control Science Center of Excellence, Air Force Res. Lab., Wright-Patterson AFB, OH 45433 david.casbeer@wpafb.af.mil

R. Beard is with the Department of Electrical Eng., Brigham Young University, Provo, Utah, 84602 beard@byu.edu

II. Task Assignment Problem

Consider the multi-agent multi-task assignment problem where a group of N_a agents attempt to service N_t tasks while trying to maximize a reward. This can be stated as:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{N_a}\sum_{k=1}^{N_t} c_{jk}(\mathbf{x}_j, \mathbf{p}_j)x_{jk} \\ \text{subject to} \quad & \sum_{j=1}^{N_a} x_{jk} \leq 1 \quad \forall k \in \{1,\dots,N_t\} \\ & \sum_{k=1}^{N_t} x_{jk} \leq L_m \quad \forall j \in \{1,\dots,N_a\}, \end{aligned} \tag{1}$$

where $x_{jk} \in \{0,1\}$ is one if agent j services task k and zero otherwise, $\mathbf{x}_j \in \{0,1\}^{N_t}$ is a vector whose kth element is x_{jk}, $\mathbf{p}_j$ is the path for agent j and indicates the order agent j will service its assigned tasks, $c_{jk}(\cdot)$ is the non-negative reward, and L_m is the maximum number of tasks per agent.

Due to the curse of dimensionality, an exact solution to the multi-agent multi-task problem becomes impractical for large numbers of agents [5]. To overcome this problem, current operations have multiple operators controlling either one UAV or possibly a group of UAVs. Each group of operators would service tasks independently. In the setup we propose, a human remains responsible for his or her team of agents. The teams interact, with other teams, through higher level coordination. Here the N_t tasks and N_a agents are divided between N_o operators. Each team $i \in \{1,\dots,N_o\}$ attempts to service all of its assigned tasks $\mathcal{N}_{ti} \subset \{1,\dots,N_t\}$ with its assigned agents $\mathcal{N}_{ui} \subset \{1,\dots,N_a\}$. The team assignment problem can be stated as:

$$\begin{aligned} \text{maximize} \quad & \sum_{j\in\mathcal{N}_{ui}}\sum_{k\in\mathcal{N}_{ti}} c_{jk}(\mathbf{x}_j, \mathbf{p}_j)x_{jk} \\ \text{subject to} \quad & \sum_{j\in\mathcal{N}_{ui}} x_{jk} \leq 1 \quad \forall k \in \mathcal{N}_{ti} \\ & \sum_{j\in\mathcal{N}_{ti}} x_{jk} \leq L_m \quad \forall j \in \mathcal{N}_{ui} \end{aligned} \tag{2}$$

A problem occurs when a team is unable to perform all of its assigned tasks due to time constraints, resource constraints, or agent capabilities. We present a hierarchical

task assignment method, that allows neighboring teams to request and provide help when this situation occurs.

III. Multi-Team Task Assignment

The Team Consensus-Based Bundle Algorithm (TCBBA) consists of two main stages: local task assignment and task sharing. In the first stage, each team implements the traditional CBBA to create an initial task assignment. If there are tasks that were unable to be assigned, the teams attempt to correct this by implementing the CBBA between the teams.

A. Local Task Assignment

To create the initial task assignment, each team runs the CBBA introduced in [1]. This description of the CBBA follows the explanation given in [2]. The CBBA is a decentralized task assignment algorithm which consists of iterations between two phases: a bundle building phase where each agent generates an ordered bundle of tasks and a consensus phase where conflicting assignments are resolved through communication with nearby agents.

1) Phase 1 (Build Bundle): The first phase of the CBBA is bundle construction, during which each agent adds tasks to its bundle until it is unable to add any more tasks. Each agent j in team i maintains four data vectors: a bundle $\mathbf{b}_{ij} \in (\mathcal{N}_{ti} \cup \{\emptyset\})^{L_m}$, the corresponding path $\mathbf{p}_{ij} \in (\mathcal{N}_{ti} \cup \{\emptyset\})^{L_m}$, a winning bid list $\mathbf{y}_{ij} \in \Re_+^{N_{ti}}$, and a winning agent list $\mathbf{z}_{ij} \in \mathcal{U}^{N_{ti}}$. Tasks in the bundle are ordered by decreasing marginal scores, while those in the path are ordered based on the order they will be performed. The marginal score of a task is defined as follows: Let $S_{ij}(\mathbf{p}_{ij})$ be defined as the total reward for agent j in team i performing the tasks within $\mathbf{p}_{ij}$. If a task k is added to the bundle $\mathbf{b}_{ij}$, it has the marginal score improvement $c_{ijk}(\mathbf{b}_{ij}) = \max_{n \leq |\mathbf{p}_{ij}|} S_{ij}(\mathbf{p}_{ij} \oplus_n \{k\}) - S_{ij}(\mathbf{p}_{ij})$ where $|\cdot|$ is the cardinality of the list and $\oplus_n$ is the operation that inserts the second list after the n-th element of the first list. The bundle is updated by $\mathbf{b}_{ij} = \mathbf{b}_{ij} \oplus_{|\mathbf{b}_{ij}|} \{K_{ij}\}$ with $K_{ij} = \text{argmax}_k \left(c_{ijk}(\mathbf{b}_{ij}) \times \prod(c_{ijk} > y_{ijk}) \times h_{ijk}\right)$ where $\prod(\cdot)$ is one if the argument is true and zero otherwise and h_{ijk} is one if agent j can perform task k and zero otherwise. The path is updated by $\mathbf{p}_{ij} = \mathbf{p}_{ij} \oplus_{n_{ijK_{ij}}} \{K_{ij}\}$ where $n_{ijK_{ij}} = \text{argmax}_n \left(S_{ij}^{\mathbf{p}_{ij} \oplus_n \{K_{ij}\}}\right)$.

2) Phase 2 (Conflict Resolution): In the conflict resolution phase, three data vectors are transmitted to neighboring agents. Two were used in the bundle construction phase: the winning bid list $\mathbf{y}_{ij}$ and the winning agent list $\mathbf{z}_{ij}$. The third data vector $\mathbf{s}_{ij} \in \Re^{N_{ai}}$ represents the time stamp of the last communication from each of the other agents within the same team. When agent j receives a message from agent m, it uses $\mathbf{z}_{ij}, \mathbf{s}_{ij}, \mathbf{z}_{im}$, and $\mathbf{s}_{im}$ to determine which agent's information is the most up to date for each task. There are three possible actions agent j can perform on task k: (1) update: $y_{ijk} = y_{imk}, z_{ijk} = z_{imk}$; (2) reset: $y_{ijk} = 0, z_{ijk} = \emptyset$; and (3) leave: $y_{ijk} = y_{ijk}, z_{ijk} = z_{ijk}$.

B. Task Sharing

After the initial plan has been created, the teams communicate with each other to determine if other teams can perform some of their tasks. This phase of the TCBBA has two variations. In the first version, the initial plan developed by the CBBA is fixed and each team attempts to insert the unassigned tasks into the initial plan. In the second version, the initial plan is only used to determine which tasks were able to be assigned. The plan itself is ignored. In both versions each team is required to maintain the same data vectors. Furthermore, both versions of the TCBBA have three stages: initialization, creating the bundle, and communication and conflict resolution. The initialization stage is the only stage that is different between the two variations of the TCBBA.

1) Version 1: Insert in Path Initialization: This version of the TCBBA keeps the initial plan that was developed by the CBBA. Algorithm 1 describes how all of the data structures are initialized to accomplish this. After the initial plan has been created, each team identifies which of its allocated tasks were assigned to one of its agents, $\tilde{\mathcal{N}}_{ti}$, and which of its tasks were unassigned, $\hat{\mathcal{N}}_{ti} = \mathcal{N}_{ti} \setminus \tilde{\mathcal{N}}_{ti}$. Each team shares its list of unassigned tasks to come up with the complete list of unassigned tasks $\mathcal{T}^u = \left(\hat{\mathcal{N}}_{t1} \cup \hat{\mathcal{N}}_{t2} \cup \ldots \cup \hat{\mathcal{N}}_{tN_o}\right)$ (A1 : 1) [1]. The team bundle $\mathbf{B}_i \in (\mathcal{T}^u \cup \{0\})^{L_m}$, winning bid list $\mathbf{Y}_i \in \Re_+^{|\mathcal{T}^u|}$, and winner ID list $\mathbf{Z}_i \in \mathcal{U}^{|\mathcal{T}^u|}$ are initialized to empty values because they only include information on tasks in $\mathcal{T}^u$ (A1 : 3-5). The team time-stamp vector, $\mathbf{S}_i$, is initialized to zeros. In addition, all of the agents' data vectors are enlarged while maintaining the same information to account for the unassigned tasks (A1 : 6-13).

After the initialization is done, it iterates between the bundle construction stage, described in Algorithm 3, and the communication and conflict resolution stage, described in Algorithms 4 and 5, until consensus has been achieved (A1 : 14-17).

Keeping the initial plan ensures that the final plan is relatively close to the original one. During the build bundle phase (Algorithm 3), the CBBA will attempt to insert the unassigned tasks into the plan without disrupting the current plan. The task will only be inserted if all of the tasks later on in the path can still be completed. This guarantees that any task that was assigned during the initial assignment will still be assigned in the final plan.

2) Version 2: Create New Plan Initialization: Modifying Algorithm 1 to ignore the initial plan is very straightforward. The data vectors $\mathbf{p}_{ij}, \mathbf{b}_{ij}, \mathbf{y}_{ij}$ and $\mathbf{z}_{ij}$ are initialized to empty or zero (A2 : 2-5) instead of the initial plan (A1 : 8-9,12-13).

This version of the TCBBA does not guarantee that all the tasks that were previously assigned in the initial assignment will be completed. Instead, this version will assign tasks that will maximize its score, even if it has to not perform some previously assigned tasks. The score can be improved further by running Algorithm 2 multiple times recomputing

[1] We refer to line 1 of Algorithm 1 as (A1 : 1). This notation will be used throughout the rest of the paper.

Algorithm 1: TCBBA version 1 initialization for team i

input : $\hat{\mathbf{b}}_i, \hat{\mathbf{p}}_i, \hat{\mathbf{y}}_i, \hat{\mathbf{z}}_i, \mathcal{N}_{ti}$
output: $\mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i, \mathbf{b}_i, \mathbf{p}_i, \mathbf{y}_i, \mathbf{z}_i$

1: $\mathcal{T}^u = \text{unassignedTasks}(\mathbf{b}_i, \mathcal{N}_{ti})$
2: $\mathcal{T}_i = \mathcal{N}_{ti} \setminus \mathcal{T}^u$
3: $\mathbf{B}_i = \{\}$
4: $\mathbf{Y}_i = \text{zeros}(|\mathcal{T}^u|)$
5: $\mathbf{Z}_i = \text{zeros}(|\mathcal{T}^u|)$
6: **foreach** $j \in \mathcal{N}_{ui}$ **do**
7: $\quad \mathbf{b}_{ij} = \hat{\mathbf{b}}_{ij}$
8: $\quad \mathbf{p}_{ij} = \hat{\mathbf{p}}_{ij}$
9: $\quad \mathbf{y}_{ij} = \text{zeros}(|\mathcal{T}_i \cup \mathcal{T}^u|)$
10: $\quad \mathbf{z}_{ij} = \text{zeros}(|\mathcal{T}_i \cup \mathcal{T}^u|)$
11: $\quad \mathbf{y}_{ijk} = \hat{\mathbf{y}}_{ijk} \quad \forall k \in \mathcal{T}_i$
12: $\quad \mathbf{z}_{ijk} = \hat{\mathbf{z}}_{ijk} \quad \forall k \in \mathcal{T}_i$
13: **end**
14: **while** *no consensus* **do**
15: $\quad [\mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i, \mathbf{b}_i, \mathbf{p}_i, \mathbf{y}_i, \mathbf{z}_i] = \text{buildBundle}(\mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i, \mathcal{T}^u, \mathbf{b}_i, \mathbf{p}_i, \mathbf{y}_i, \mathbf{z}_i, \mathcal{T}_i)$
16: $\quad [\mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i, \mathbf{b}_i, \mathbf{p}_i, \mathbf{y}_i, \mathbf{z}_i] = \text{communicate}(\mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i, \mathbf{b}_i, \mathbf{p}_i, \mathbf{y}_i, \mathbf{z}_i)$
17: **end**

Algorithm 2: Changes to Algorithm 1 for TCBBA version 2 initialization

1: **foreach** $j \in \mathcal{N}_{ui}$ **do**
2: $\quad \mathbf{b}_{ij} = \{\}$
3: $\quad \mathbf{p}_{ij} = \{\}$
4: $\quad \mathbf{y}_{ij} = \text{zeros}(|\mathcal{T}_i \cup \mathcal{T}^u|)$
5: $\quad \mathbf{z}_{ij} = \text{zeros}(|\mathcal{T}_i \cup \mathcal{T}^u|)$
6: **end**

the unassigned tasks each time. This allows the teams to trade tasks to improve the overall score in exchange for more computation time and communication steps.

C. Phase 1: Build Bundle

First the bids for the unassigned tasks and plans for each agent must be computed. This is done by running a CBBA between all of the agents within the team (A3 : 1). After the agent bundles have been built, the information must be extracted from the agents and put into the team data structures. To do this it goes through each agent's bundle looking for tasks $\{k\} \in \mathcal{T}^u$ and putting them in the team's bundle $\mathbf{B}_i$ in order of decreasing bid (A3 : 6-12). Sorting the bundle is necessary to satisfy the diminishing marginal gain requirement mentioned in [1]. The winning bid list is set to the agent winning bid and the winner ID list is set to i (A3 : 13-14).

D. Phase 2: Communication

The communication phase consists of three steps: sending and receiving data, resolving conflicts in the team's bundle,

Algorithm 3: TCBBA build bundle for team i

input : $\mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i, \mathcal{T}^u, \hat{\mathbf{b}}_i, \hat{\mathbf{p}}_i, \hat{\mathbf{y}}_i, \hat{\mathbf{z}}_i, \mathcal{T}_i$
output: $\mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i, \mathbf{b}_i, \mathbf{p}_i, \mathbf{y}_i, \mathbf{z}_i$

1: $[\mathbf{b}_i, \mathbf{p}_i, \mathbf{y}_i, \mathbf{z}_i] = \text{CBBA}(\hat{\mathbf{b}}_i, \hat{\mathbf{p}}_i, \hat{\mathbf{y}}_i, \hat{\mathbf{z}}_i, \mathcal{T}_i \cup \mathcal{T}^u)$
2: **foreach** $j \in \mathcal{N}_{ui}$ **do**
3: $\quad$ **for** $k = 1$ **to** $|\mathbf{b}_{ij}|$ **do**
4: $\quad\quad$ **if** $\mathbf{b}_{ijk} \in \mathcal{T}^u$ **then**
5: $\quad\quad\quad n = |\mathbf{B}_i|$
6: $\quad\quad\quad$ **for** $m = |\mathbf{B}_i|$ **to** 1 **do**
7: $\quad\quad\quad\quad$ **if** $\mathbf{Y}_{i\mathbf{B}_{im}} < \mathbf{y}_{ij\mathbf{b}_{ijk}}$ **then**
8: $\quad\quad\quad\quad\quad n = m - 1$
9: $\quad\quad\quad\quad$ **end**
10: $\quad\quad\quad\quad \mathbf{B}_i = \mathbf{B}_i \oplus_n \mathbf{b}_{ijk}$
11: $\quad\quad\quad$ **end**
12: $\quad\quad\quad \mathbf{Z}_{i\mathbf{b}_{ijk}} = i$
13: $\quad\quad\quad \mathbf{Y}_{i\mathbf{b}_{ijk}} = \mathbf{y}_{ij\mathbf{b}_{ijk}}$
14: $\quad\quad$ **end**
15: $\quad$ **end**
16: **end**

and resolving conflicts in the agents' bundles. For team i, the team sends $\mathbf{Y}_i, \mathbf{Z}_i$, and $\mathbf{S}_i$ and receives $\mathbf{Y}_h, \mathbf{Z}_h$, and $\mathbf{S}_h$ from team h. It iterates through each task and determines what action to take. The possible actions are the same used in the CBBA and are repeated in Table I. There are three possible outcomes from the decision table (A4 : 6): update, reset, or leave. An update for team i means team i's information is changed to what team h knows (A4 : 8). A reset re-initializes the bid and winner ID to zero (A4 : 10), and the task is removed from the bundle (A4 : 15-16). A leave makes no change to any of the data vectors. If an update or a reset occurred on task k then the reward values for the tasks that appear in the bundle after task k are no longer valid and all of the following tasks in the bundle need to be reset (A4 : 15-16). The time stamp vector is then updated using the current time and $\mathbf{S}_j$ (A4 : 21-27).

After the team information $\mathbf{B}_i, \mathbf{Y}_i$, and $\mathbf{Z}_i$ has been updated, the individual agent's information needs to be updated (A4 : 26-28). Each agent $j \in \mathcal{N}_{ui}$ goes through its entire bundle $\mathbf{b}_{ij}$ looking for any tasks that were removed from the team's bundle (A5 : 1-2). If it finds one then, for that task and each task that follows it in the bundle, the bid and the winner ID are reset to zero (A5 : 3) and the tasks are removed from the bundle and the path (A5 : 4-5).

IV. Convergence Time Analysis

It is informative to determine the maximum number of communication steps required for all teams to arrive at a conflict free solution in a static communication network and worst case scenario. We cannot perform this analysis on Algorithm 2 requires knowledge of the number of tasks assigned by each team during each iteration.

In the worst case situation, the diameter of the communication network equals the number of agents. Let D_i be

Algorithm 4: Conflict resolution for team i receiving from team j at time τ

input : $\mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i, \mathcal{T}^u$
output: $\mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i$

SEND: $\mathbf{Y}_i, \mathbf{Z}_i, \mathbf{S}_i$ to team j
RECEIVE: $\mathbf{Y}_j, \mathbf{Z}_j, \mathbf{S}_j$ from team j
foreach $k \in \mathcal{T}^u$ **do**
  $\hat{Y}_i = \mathbf{Y}_{ik}, \ \hat{Z}_i = \mathbf{Z}_{ik}$
  $\hat{Y}_j = \mathbf{Y}_{jk}, \ \hat{Z}_j = \mathbf{Z}_{jk}$
  Find action (Table I) $\Rightarrow$ UPDATE, RESET, or LEAVE $(\hat{Y}_i, \hat{Y}_j, \hat{Z}_i, \hat{Z}_j, \mathbf{S}_i, \mathbf{S}_j)$
  if *UPDATE* **then**
    $\mathbf{Y}_{ik} = \hat{Y}_j, \ \mathbf{Z}_{ik} = \hat{Z}_j$
  else if *RESET* **then**
    $\mathbf{Y}_{ik} = 0, \quad \mathbf{Z}_{ik} = 0$
  end
  if *UPDATE* **or** *RESET* **then**
    for $m = 1$ **to** $|\mathbf{B}_i|$ **do**
      if $\mathbf{B}_{im} = k$ **then**
        $\mathbf{Y}_{i,\mathbf{B}_{in}} = 0, \ \mathbf{Z}_{i,\mathbf{B}_{in}} = 0,$
        $\mathbf{B}_{in} = \emptyset \quad \forall m \leq n \leq L_m$
      end
    end
  end
end
for $k = 1$ **to** N_o **do**
  if $k = j$ **then**
    $\mathbf{S}_{ik} = \tau$
  else
    $\mathbf{S}_{ik} = \max(\mathbf{S}_{ik}, \mathbf{S}_{jk})$
  end
end
foreach $j \in \mathcal{N}_{ui}$ **do**
  $\text{UpdateAgentList}(j, \mathbf{B}_i, \mathbf{Y}_i, \mathbf{Z}_i)$
end

the diameter of the communication network of the agents in team i, N_i be the number of tasks assigned to team i, D_T be the diameter of the team communication network, N_u be the total number of unassigned tasks after the local task assignment phase, N be the total number of tasks, and D be the diameter of the communication network of all the agents.

From [1] we know that the maximum number of communication steps needed for the CBBA to converge to a conflict free solution is ND. Applying this result to the TCBBA, the number of communications steps needed in the worst case for the initial task assignment is $\max_i(N_i D_i)$. The second stage will have at most $N_u D_T$ iterations and each iteration will have $\max_i N_u D_i$ communication steps between agents as well as one communication step between teams. The total

TABLE I

ACTION RULE FOR AGENT i RECEIVING FROM AGENT j [1]

Sender's $\hat{z}_j$	Receiver's $\hat{z}_i$	Condition(s)	Action (Default: LEAVE)
j	i	if $\hat{y}_j > \hat{y}_i$	UPDATE
	j or 0		UPDATE
	$\eta \notin \{i, j\}$	if $\mathbf{S}_{j\eta} > \mathbf{S}_{i\eta}$ or $\hat{y}_j > \hat{y}_i$	UPDATE
i	i or 0		LEAVE
	j		RESET
	$\eta \notin \{i, j\}$	if $\mathbf{S}_{j\eta} > \mathbf{S}_{i\eta}$	RESET
$\mu \notin \{i, j\}$	i	$\mathbf{S}_{j\mu} > \mathbf{S}_{i,\mu}$ and $\hat{y}_j > \hat{y}_i$	UPDATE
	j	if $\mathbf{S}_{j\mu} > \mathbf{S}_{i\mu}$	UPDATE
		else	RESET
	μ or 0	if $\mathbf{S}_{j\mu} > \mathbf{S}_{i\mu}$	UPDATE
	$\eta \notin \{i, j, \mu\}$	if $\mathbf{S}_{j\mu} > \mathbf{S}_{i\mu}$ and $\mathbf{S}_{j\eta} > \mathbf{S}_{i\eta}$	UPDATE
		if $\mathbf{S}_{j\mu} > \mathbf{S}_{i\mu}$ and $\hat{y}_j > \hat{y}_i$	UPDATE
		if $\mathbf{S}_{j\eta} > \mathbf{S}_{i\eta}$ and $\mathbf{S}_{i\mu} > \mathbf{S}_{j\mu}$	RESET
0	i or 0		LEAVE
	j		UPDATE
	$\eta \notin \{i, j\}$	if $\mathbf{S}_{j\eta} > \mathbf{S}_{i\eta}$	UPDATE

Algorithm 5: Conflict resolution for team i agent j

input : $\mathbf{B}_i, \mathcal{T}^u, \mathbf{b}_{ij}, \mathbf{p}_{ij}, \mathbf{y}_{ij}, \mathbf{z}_{ij}$
output: $\mathbf{b}_{ij}, \mathbf{p}_{ij}, \mathbf{y}_{ij}, \mathbf{z}_{ij}$

for $k = 1$ **to** $|\mathbf{b}_{i,j}|$ **do**
  if $\mathbf{b}_{ijk} \in \mathcal{T}^u$ **and** $\mathbf{b}_{ijk} \notin \mathbf{B}_i$ **then**
    $\mathbf{y}_{i,j,\mathbf{b}_{ijm}} = 0, \ \mathbf{z}_{i,j,\mathbf{b}_{ijm}} = 0 \quad \forall k \leq m \leq |\mathbf{b}_{ij}|$
    $\mathbf{b}_{ijm} = \emptyset \quad \forall k \leq m \leq |\mathbf{b}_{ij}|$
    Update $\mathbf{p}_{ij}$: For each task that was removed from $\mathbf{b}_{ij}$ remove it from $\mathbf{p}_{ij}$
  end
end

number of communication steps is

$$N_c = \max_i(N_i D_i) + D_T N_u \left(\max_i(N_u D_i) + 1\right). \quad (3)$$

In the worst case, $\max_i(N_u D_i) \gg 1$ so

$$N_c \approx \max_i(N_i D_i) + D_T N_u^2 \max_i(D_i). \quad (4)$$

If we assume that the agents and tasks are divided up equally among the N_o teams then

$$N_i = \frac{N}{N_o} \quad \text{and} \quad D_i = \frac{D}{N_o}.$$

and (4) becomes

$$N_c = \frac{ND}{N_o^2} + \frac{D_T N_u^2 D}{N_o}. \quad (5)$$

If we assume worst case communication between the teams then $N_o = D_T$ and (5) becomes

$$\frac{ND}{N_o^2} + N_u^2 D = \left(\frac{N}{N_o^2} + N_u^2\right) D. \quad (6)$$

Comparing (6) to the worst case convergence bound for the full CBBA assignment ND we find

$$\left(\frac{N}{N_o^2} + N_u^2\right) D \leq ND. \quad (7)$$

Solving (7) for N_u we obtain

$$N_u \leq \sqrt{1 - \frac{1}{N_o^2}}\sqrt{N}. \quad (8)$$

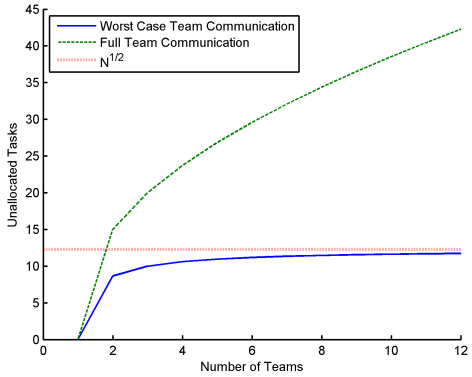

Fig. 1. Maximum number of unassigned tasks for TCBBA to converge quicker than CBBA in worst case as the number of teams changes ($N_t = 150, D = 12$).

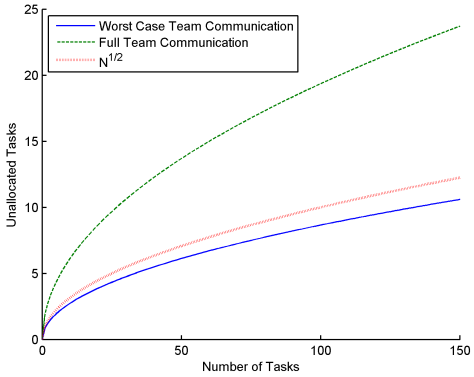

Fig. 2. Maximum number of unassigned tasks for TCBBA to converge quicker than CBBA in worst case as the number of tasks changes ($N_o = 4, D = 12$).

Notice as the number of teams increases, the number of allowed unassigned tasks, N_u approaches $\sqrt{N}$.

If we assume that there is full communication between the teams instead of worst case then $D_T = 1$ and (5) becomes

$$\frac{ND}{N_o^2} + \frac{N_u^2 D}{N_o} = \left(\frac{N}{N_o^2} + \frac{N_u^2}{N_o}\right) D. \tag{9}$$

Comparing (9) to the worst case convergence bound for the full CBBA assignment ND we find

$$\left(\frac{N}{N_o^2} + \frac{N_u^2}{N_o}\right) D \leq ND. \tag{10}$$

Solving (10) for N_u we obtain

$$N_u \leq \sqrt{N_o - \frac{1}{N_o}}\sqrt{N}. \tag{11}$$

Figure 1 shows how the maximum number of unassigned tasks changes as the number of teams increase while the number of tasks and agents stay the same. Notice in the worst case team communication case, the maximum number of unassigned tasks quickly approaches the maximum value of $\sqrt{N_t}$. In the full team communication case, the number of unassigned tasks quickly exceeds $\sqrt{N_t}$.

Figure 2 shows how the maximum number of unassigned tasks changes as the number of tasks increase while the number of teams and agents stay the same. Notice that both the worst case team communication and full team communication increase proportionally to $\sqrt{N_t}$.

V. Simulation Results

We created four scenarios in order to compare the TCBBA to the CBBA. In each scenario there are three teams of four agents within a 1200x400 meter world. Each team is assigned N_u tasks which take 25 seconds to complete. There are four types of tasks $\{A,\ B,\ C,\ D\}$ and four types of agents $\{a,\ b,\ c,\ d\}$. Each agent type is only able to service its corresponding task type, moves at $40m/s$, and has no turning constraints. We used a time-discounted reward for the tasks:

$$S_i^{\mathbf{p}_i} = \sum \lambda_j^{\tau_i^j(\mathbf{p}_i)} c_j \tag{12}$$

where $S_i^{\mathbf{p}_i}$ is the total score of agent i with path $\mathbf{p}_i$, $\lambda_j = 0.001$ is the discounting factor for task j, $\tau_i^j(\mathbf{p}_i)$ is the estimated time that agent i will service task j when following path $\mathbf{p}_i$, and $c_j = 100$ is the value of the task. The objective is the maximize the total score over 500 seconds.

The four scenarios are created by changing two parameters. The first parameter is if the teams have their own region of the map or if they share the entire world. If the teams are not overlapping, then they each have their own 400x400 meter region with team two located in the center. If the teams are overlapping then the teams' agents and tasks are randomly distributed throughout the entire world. The second parameter is if there are 50 or 100 tasks per team.

To force the teams to cooperate, teams one and three are not given a complete set of agents. Team one has two type a agents and two type b agents, team two has one of each type of agent, and team three has two type c agents and two type d agents. Each team is assigned an equal number of each task type which are randomly distributed within the team's assigned region. Providing an incomplete set of agents to teams one and three guarantees there will be unassigned tasks after the initial CBBA. It also guarantees that the proportion of unassigned tasks to total tasks will be higher than the threshold developed in the previous section implying that the TCBBA will take more communication steps than the CBBA.

Five hundred Monte-Carlo simulations are run for each scenario. In each simulation, we assign the tasks five different ways: running Algorithm 1 once, running Algorithm 2 once, running Algorithm 2 until the change in score is less than 2.5%, running Algorithm 2 until either all the tasks have been assigned or the unassigned tasks list stops changing, and merging all the tasks and agents into one team and running the CBBA. Figure 3 shows the mean percent difference in total score between the four TCBBA versions and the CBBA assignment. Figure 4 shows the mean percent difference in communication steps needed to arrive at a conflict free solution between the TCBBA and the CBBA. Figure 5 shows the mean percent difference in computation time between the

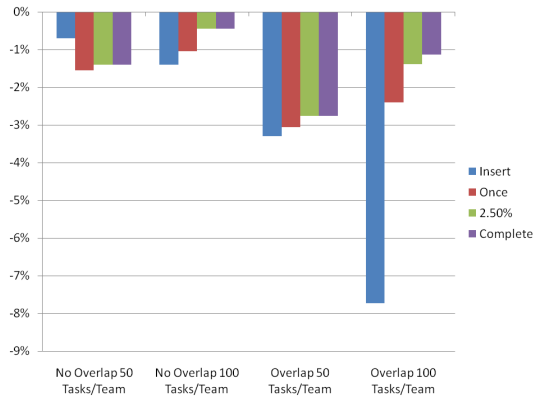

Fig. 3. Mean percent difference in overall score between TCBBA and CBBA

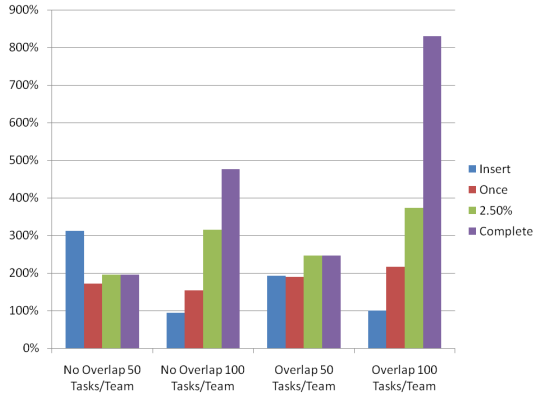

Fig. 4. Mean percent difference in communication steps between TCBBA and CBBA

TCBBA and CBBA.

The results show several interesting things about the TCBBA. First, as seen in Figure 3, all the variations of the TCBBA consistently have a lower overall score than the CBBA. This difference varies based on the which TCBBA variation is used but is typically within 3%. What is not shown is that occasionally the TCBBA can do as well as the CBBA but in all five hundred runs, it never beat it. Second, the TCBBA always required more communication steps than the CBBA as shown in Figure 4. This result was expected because of the large number of unassigned tasks. Third, repeating the second variation of TCBBA until the unassigned task list is empty or unchanging is not worth the computation. Finally, the amount of computation time required is proportional to the number of communication steps as seen in Figure 5. It is interesting to note that the TCBBA can take less time to arrive at a solution than the CBBA when implemented in a centralized manner.

VI. Conclusion and Future Work

This paper presents a modification of the Consensus-Based Bundle Algorithm that handles the multi-team, multi-task problem using a combination of auctions and consensus to

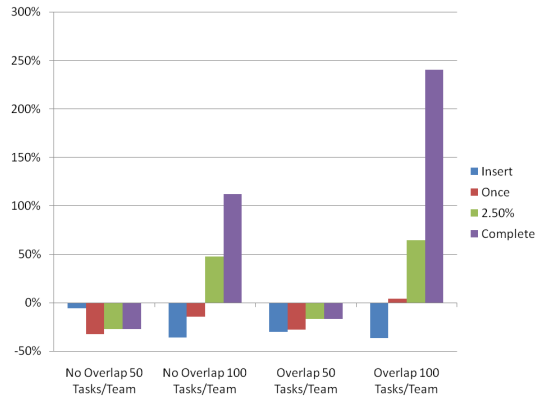

Fig. 5. Mean percent difference in computation time between TCBBA and CBBA

achieve feasible, conflict free solutions. While these solutions are less optimal than running the CBBA, typically within 3%, the TCBBA allows the agents or region to be divided into separate teams to handle a wider variety of situations.

In the future, we want to modify Algorithm 2 so that it no longer ignores the initial plan. Instead it will use the approach taken by the Extended Consensus-Based Bundle Algorithm and will attempt to insert the unassigned tasks into the initial plan while still allowing the initial plan to be modified as described in [6]. Furthermore, we want to allow teams to be able to ask for help from neighboring teams instead of from all teams. These changes should drastically reduce the number of communication steps required to achieve consensus while decreasing the final score by a small amount.

VII. Acknowledgments

We would like to thank Jonathan How and Sameera Ponda for allowing us to use their CBBA Matlab code.

References

[1] Han-Lim Choi, Lue Brunet, and Jonathan P How. Consensus-Based Decentralized Auctions for Robust Task Allocation. *Robotics, IEEE Transactions on*, 25(4):912 – 926, 2009.

[2] Han-Lim Choi, Andrew K Whitten, and Jonathan P How. Decentralized Task Allocation for Heterogeneous Teams with Cooperation Constraints. In *American Control Conference*, pages 3057–3062, 2010.

[3] Sameera Ponda, Josh Redding, Han-lim Choi, Jonathan P How, Matt Vavrina, and John Vian. Decentralized Planning for Complex Missions with Dynamic Communication Constraints. In *American Conrol Conference*, pages 3998–4003, 2010.

[4] Sameera Ponda, Han-lim Choi, and Jonathan P How. Predictive planning for heterogeneous teams with human agents. In *AIAA Infotech@Aerospace*, 2010.

[5] Richard Ernest Bellman. *Dynamic Programming*. Rand Corporation, 1957.

[6] Travis Mercker, David W Casbeer, P Travis Millet, and Maruthi R Akella. An Extension of Consensus-Based Auction Algorithms for Decentralized , Time-Constrained Task Assignment. In *American Control Conference*, 2010.