Decentralized task allocation for heterogeneous teams with cooperation constraints

2010 American Control Conference
Marriott Waterfront, Baltimore, MD, USA
June 30-July 02, 2010

Decentralized Task Allocation for Heterogeneous Teams with Cooperation Constraints

Han-Lim Choi, Andrew K. Whitten, and Jonathan P. How

Abstract— **This paper presents decentralized methods for allocating heterogeneous tasks to a network of agents with different capabilities, when the rules of engagement dictate various cooperation constraints. The new methods are built upon the consensus-based bundle algorithm (CBBA), and the key extensions to the baseline CBBA are: (a) task decomposition and associated scoring modification to allow for soft-constrained cooperation preference, and (b) a decentralized task elimination protocol to ensure satisfaction of the hard-constrained cooperation requirements. The new extensions are shown to preserve the robust convergence property of the baseline CBBA, and numerical examples on a cooperative track & strike mission verify the performance improvement by these extended capabilities.**

I. Introduction

Decentralized task planning for a team of networked agents (e.g., unmanned aerial vehicles (UAVs)) has been investigated by many researchers to overcome the possibility of single-points of failure and mission range restrictions of centralized planning [1]–[6]. Some decentralized methods instantiate the centralized planner on each agent [7]–[10], and use information consensus to ensure that all the agents agree upon the information set used for formulation of individual instances of the centralized planning problem. However, this consensus process can be slow, and defining a sufficient level of consensus remains an open question, since slight differences in information can lead to inconsistent and conflicting decisions [11,12].

As an alternative approach, the authors have recently developed the consensus-based bundle algorithm (CBBA) [13] that is a market-based distributed agreement protocol upon the winning agents and associated winning scores. By enforcing agreement upon the solution rather than the information set, CBBA was shown to account for inconsistent information among the networked agents, guaranteeing a conflict-free assignment for all the agents in the network. CBBA is a polynomial-time algorithm that scales well with the size of the network and/or the number of tasks (or equivalently, the length of the planning horizon). From the design perspective, various design objectives, agent models, and constraints can be incorporated by defining appropriate scoring functions. If the resulting scoring scheme satisfies a property called *diminishing marginal gain* (DMG), a provably good feasible, i.e., conflict-free, solution is guaranteed.

H.-L. Choi, A. K. Whitten, and J. P. How was/are with the Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA. {hanlimc, awhitten, jhow}@mit.edu. H.-L. Choi is currently with the Div. of Aerospace Engineering, KAIST, Korea.

While CBBA has provided a basic framework to allocate a given set of tasks fixed in location to a specified set of agents at fixed locations, addressing complex cooperative missions in uncertain, dynamic environments may require a type of problem formulation that the baseline CBBA was not originally designed to handle. Other papers by the authors have addressed time windows of validity for tasks and dynamic network topology during the mission [14], and incorporation of obstacle fields and mitigation of chattering in plans due to noisy situational awareness [15].

This paper particularly focuses on constraints due to requirements and preferences of cooperation. In missions with heterogeneous networked agents *rules of engagement* often dictate on which types of agents, and how many of each are needed to complete a given type of task. To address these requirements, this work first presents a framework to embed the cooperation constraints in the scoring structure; then, decentralized methods to eliminate invalid assignment, i.e., assignment violating the requirements, are proposed. Moreover, to allow for handling of soft-constrained cooperation preference, this paper relaxes the restriction on the number of agents per task that existed in the baseline CBBA. The proposed methodologies are shown to preserve robust convergence properties of CBBA. Numerical examples of cooperative track and destroy mission verify the proposed algorithms.

II. Problem Statement

A. Task Allocation with Duo Cooperation

The goal of heterogeneous task allocation is, given a list of N_t tasks of K types and a list of N_a agents of M types, to find a conflict-free matching of tasks to a required set of agents that maximizes some global reward. This paper particularly considers the case where the network consists of two different types of agents, denoted as $\mathcal{I}_1$ and $\mathcal{I}_2$, and there are three different types of tasks described as follows:

a) Solo Task (ST): a task that is completed by a single agent and assignment of additional agents is unacceptable (or in a more relaxed sense, non-beneficial). Depending on the capability characteristics of the agents, some solo tasks must exclusively be performed by a certain type, and some can be performed by either type. For notational simplicity, this paper does not distinguish the exclusive case and the non-exclusive case, as this can be represented by appropriately defining the reward structure. $\mathcal{J}_S$ denotes the set of STs.

b) Preferred Duo Task (PDT): A task that can be done by either one or two agents. Assignment of two agents typically results in greater reward than the assignment of

single agents, but assignment of three or more agents is not allowed. Depending on the capability, some tasks of this type require assignment of agents of a specified type. A PDT can be treated as a generalization of ST to allow assignment of two agents. $\mathcal{J}_{PD}$ denotes the set of PDTs.

c) Required Duo Task (RDT): A task that must be done by cooperation of one agent from type 1 and one agent from type 2. Assignment of single tasks does not provide any reward. Assignment of three or more agents is not allowed. $\mathcal{J}_{RD}$ denotes the set of RDTs.

For the heterogeneous network of two types, the allocation of heterogenous tasks of the aforementioned category can be written as the following optimization:

$$\max \sum_{j\in\mathcal{J}_S}\sum_{i\in\mathcal{I}} c_{ij}(\mathbf{p}_i)x^s_{ij} + \sum_{j\in\mathcal{J}_{PD}}\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{I}} d^p_{ikj}(\mathbf{p}_i,\mathbf{p}_k)x^p_{ikj} + \sum_{j\in\mathcal{J}_{RD}}\sum_{i\in\mathcal{I}_1}\sum_{k\in\mathcal{I}_2} d^r_{ikj}(\mathbf{p}_i,\mathbf{p}_k)x^r_{ikj} \quad (1)$$

subject to:

$$\forall j\in\mathcal{J}_S:\ \textstyle\sum_{i\in\mathcal{I}} x^s_{ij}\le 1, \quad (2)$$

$$\forall j\in\mathcal{J}_{PD}:\ \textstyle\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{I}} x^p_{ikj}\le 1, \quad (3)$$

$$\forall j\in\mathcal{J}_{RD}:\textstyle\sum_{i\in\mathcal{I}_1}\sum_{k\in\mathcal{I}_2} x^r_{ikj}\le 1, \quad (4)$$

$$\forall i\in\mathcal{I}_1:\ \textstyle\sum_{j\in\mathcal{J}_S} x^s_{ij} + \sum_{k\in\mathcal{I}}\sum_{j\in\mathcal{J}_{PD}}(x^p_{ikj}+\delta_{ik}x^p_{kij}) + \sum_{j\in\mathcal{J}_{RD}}\sum_{k\in\mathcal{I}_2} x^r_{ikj}\le L_t, \quad (5)$$

$$\forall i\in\mathcal{I}_2:\ \textstyle\sum_{j\in\mathcal{J}_S} x^s_{ij} + \sum_{k\in\mathcal{I}}\sum_{j\in\mathcal{J}_{PD}}(x^p_{ikj}+\delta_{ik}x^p_{kij}) + \sum_{j\in\mathcal{J}_{RD}}\sum_{k\in\mathcal{I}_1} x^r_{ikj}\le L_t, \quad (6)$$

$$x^s_{ij}\in\{0,1\},\ x^p_{ikj}\in\{0,1\},\ x^r_{ikj}\in\{0,1\}.$$

$x^s_{ij}=1$ if agent i is assigned to a solo task j and 0 otherwise, and the associated reward for this assignment is c_{ij}. It is assumed that the reward c_{ij} is uniquely determined by the path of agent i, $\mathbf{p}_i$. The path is defined as the sequence of tasks agent i performs. Decision variables x^p_{ikj}'s are employed to describe the allocation of preferred duo tasks. $x^p_{ikj}=1$ if both agent i and k are assigned to a PDT j, and this assignment incurs a reward of d^p_{ikj}, which depends on the paths of agent i and k. This work allows d^p_{ikj} and d^p_{kij} to be different from each other, to model dependency of the score on the order of task participation. Thus, the associated decision variables x^p_{ikj} and x^p_{kij} are not necessarily the same in general. In case $i=k$, $x^p_{ikj}=1$ means a single agent i, namely k, is assigned to a PDT j. As for the required duo assignment, $y^r_{ikj}=1$ if agent i of type 1 and agent k of type 2 are assigned to a RDT j, incurring reward of d^r_{ikj} that depends on the paths of agent i and k. Thus, the objective of task allocation is to maximize the sum of the rewards from assignment of STs, PDTs, and RDTs.

This maximization is subject to a set of constraints. (2) enforces a solo task to be assigned to at most one agent; (3) and (4) enforce a duo task to be assigned to at most one pair of agents (of appropriate types). Constraints in (5) and (6) limit the maximum number of tasks an agent can take up to L_t. δ_{ik} in (5) and (6) is Kronecker delta, which is introduced to avoid double counting the case of single agent being assigned to a PDT.

B. Additive Formulation for Duo Scores

This work assumes that the duo score can be decomposed into the two individual scores each agent receives upon task completion as the following assumptions:

1) PDT Score Decomposition: For the score of completing a duo task $j\in\mathcal{J}_{PD}$, there exists an equivalent scoring decomposition schemes for the participating two agents i and j such that

$$d^p_{ikj}(\mathbf{p}_i,\mathbf{p}_k) = \bar{c}^p_{ij}(\mathbf{p}_i) + \tilde{c}^p_{kj}(\mathbf{p}_k), \quad (7)$$

$$d^p_{kij}(\mathbf{p}_k,\mathbf{p}_i) = \bar{c}^p_{kj}(\mathbf{p}_k) + \tilde{c}^p_{ij}(\mathbf{p}_i) \quad (8)$$

where $\bar{c}_{ij}$ and $\tilde{c}_{ij}$ are the scores agent i can obtain by participating in task j as the roles of *leader* and *follower*, respectively. $\bar{c}_{kj}$ and $\tilde{c}_{kj}$ are defined in a similar manner. The terminology of a leader and a follower does not imply any hierarchy in the roles in performing missions, but it is employed for the purpose of easy descriptions.

To incorporate the score decomposition described above into the task allocation formulation described in section II-A, new decision variables $\bar{x}^p_{ij}\in\{0,1\},\ \forall i\in\mathcal{I}$, $\tilde{x}^p_{ij}\in\{0,1\},\ \forall i\in\mathcal{I}$ are defined such that:

$$x^p_{ikj} = \bar{x}^p_{ij}\tilde{x}^p_{kj},\ \forall i\in\mathcal{I},\ \forall k\neq i, \quad (9)$$

$$\bar{x}^p_{ij}\tilde{x}^p_{ij} = 0,\ \forall i\in\mathcal{I}. \quad (10)$$

$\bar{x}^p_{ij}=1$ if agent i takes part in completing task j as a leader, and zero otherwise. Similarly, $\tilde{x}^p_{ij}=1$ if it takes the follower role. The condition (10) ensures that no single agent takes a given task as two different roles. With these new decision variables, the second term in the objective function in (1) can be replaced by

$$\textstyle\sum_{j\in\mathcal{J}_{PD}}\sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{I}} \bar{c}^p_{ij}\bar{x}^p_{ij} + \tilde{c}^p_{kj}\tilde{x}^p_{kj},$$

and (10) is included in the constraints.

2) RDT Score Decomposition: For the score of completing a duo task $j\in\mathcal{J}_{RD}$, there exists an equivalent scoring decomposition scheme for the participating two agents $i\in\mathcal{I}_1$ and $k\in\mathcal{I}_2$ such that

$$d^r_{ikj}(\mathbf{p}_i,\mathbf{p}_k) = \hat{c}^r_{ij}(\mathbf{p}_i) + \hat{c}^r_{kj}(\mathbf{p}_k) \quad (11)$$

where $\hat{c}^r_{ij}$ and $\hat{c}^r_{kj}$ represents the contribution of agent i and agent k in completing the RDT j, respectively.

To embed these decomposed scores into the presented task allocation formulations, new binary decision variables $\hat{x}^r_{ij}\in\{0,1\},\ \forall i$ are defined to satisfy

$$x^r_{ikj} = \hat{x}^r_{ij}\hat{x}^r_{kj},\ \forall i\in\mathcal{I}_1,\ \forall k\in\mathcal{I}_2 \quad (12)$$

and the third term in (1) can be written as

$$\textstyle\sum_{j\in\mathcal{J}_{RD}}\sum_{i\in\mathcal{I}_1}\sum_{k\in\mathcal{I}_2}(c^r_{ij}\hat{x}^r_{ij} + c^r_{kj}\hat{x}^r_{kj})[\hat{x}^r_{ij}\hat{x}^r_{kj}]. \quad (13)$$

The PDT portion of the objective function can be viewed as the sum of individual scores multiplied by some indicator, $\hat{x}^r_{ij}\hat{x}^r_{kj}$ that indicates whether or not the cooperation requirement is satisfied.

III. CONSENSUS-BASED BUNDLE ALGORITHM (CBBA)

This section briefly summarizes the consensus-based bundle algorithm (CBBA) that the present authors have presented as a decentralized solution protocol for task allocation with solo tasks.

A. Algorithm

CBBA consists of iterations between two phases: a bundle building phase where each vehicle greedily generates an ordered bundle of tasks, and a consensus phase where conflicting assignments are identified and resolved through local communication between neighboring agents.

1) Phase 1 (Bundle Construction): The first phase of CBBA is bundle construction during which each agent continuously adds tasks to its bundle until it is incapable of adding any others. Each agent carries four vectors: a winning bid list $\mathbf{y}_i \in \mathbb{R}_+^{N_t}$, a winning agent list $\mathbf{z}_i \in \mathcal{I}^{N_t}$, a bundle $\mathbf{b}_i \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$, and the corresponding path $\mathbf{p}_i \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$. Tasks in the bundle are ordered by the marginal scores, while those in the path are ordered based on their assignment location. The marginal score of a task is defined as follows: let $S_i(\mathbf{p}_i)$ be defined as the total reward value for agent i performing the tasks along the path $\mathbf{p}_i$. If a task j is added to the bundle $\mathbf{b}_i$, it incurs the marginal score improvement $c_{ij}[\mathbf{b}_i] = \max_{n \leq |\mathbf{p}_i|} S_i(\mathbf{p}_i \oplus_n \{j\}) - S_i(\mathbf{p}_i)$ where $|\cdot|$ denotes the cardinality of the list, and $\oplus_n$ denotes the operation that inserts the second list right after the n-th element of the first list. The bundle is recursively updated as $\mathbf{b}_i = \mathbf{b}_i \oplus_{|\mathbf{b}_i|} \{J_i\}$with $J_i = \text{argmax}_j(c_{ij}[\mathbf{b}_i] \times \mathbb{I}(c_{ij} > y_{ij}))$, $n_{i,J_i} = \text{argmax}_n S_i^{\mathbf{p}_i \oplus_n \{J_i\}}$, and $\mathbb{I}(\cdot) = 1$ if the argument is true and zero if it is false.

2) Phase 2 (Conflict Resolution): In the conflict resolution phase, three vectors are communicated for consensus. Two were described in the bundle construction phase: the winning bids list $\mathbf{y}_i \in \mathbb{R}^{N_t}$ and the winning agent list $\mathbf{z}_i \in \mathcal{I}^{N_t}$. The third vector $\mathbf{s}_i \in \mathbb{R}^{N_u}$ represents the time stamp of the last information update from each of the other agents. When agent i receives a message from agent k, $\mathbf{z}_i$ and $\mathbf{s}_i$ are used to determine which agent's information is the most up-to-date for each task. There are three possible actions agent i can take on task j: (a) *update*: $y_{ij} = y_{kj},\ z_{ij} = z_{kj}$; (b) *reset*: $y_{ij} = 0,\ z_{ij} = \emptyset$; and (c) *leave*: $y_{ij} = y_{ij},\ z_{ij} = z_{ij}$. If a bid is changed as an outcome of communication, each agent checks if any of the updated or reset tasks were in their bundle.

B. CBBA properties

CBBA is proven to exhibit some convergence and performance properties, under the following assumption on the scoring functions: *Diminishing Marginal Gain (DMG) Assumption:*The value of a task does not increase as other elements are added to the set before it; namely,

$$c_{ij}[\mathbf{b}_i] \geq c_{ij}[\mathbf{b}_i \oplus_f \mathbf{b}] \tag{14}$$

for all $\mathbf{b}_i, \mathbf{b}, j$ such that $((\mathbf{b}_i \oplus_f \mathbf{b}) \oplus_f \{j\}) \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$ where $\emptyset$ denotes an empty task.

Provided that the scoring function is DMG, the CBBA process with a synchronized conflict resolution phase over a static communication network with diameter D satisfies the following:

1) CBBA produces the same solution as the sequential greedy algorithm (SGA), with the corresponding winning bid values and winning agent information being shared across the fleet; i.e.,
$$z_{i,j_k^\star} = i_k^\star,\ y_{i,j_k^\star} = c_{i_k^\star,j_k^\star}^{(k)},\ \forall k \leq N_{\min}$$
where $N_{\min} \triangleq \min\{N_a L_t, N_t\}$. SGA is a centralized process that sequentially solves $(i_n^\star, j_n^\star) = \text{argmax}_{(i,j) \in \mathcal{I} \times \mathcal{J}} c_{ij}^{(n)}$ with updating the scores with $c_{i,j_n^\star}^{(n+1)} = 0,\ c_{ij}^{(n+1)} = c_{ij}[\mathbf{b}_i^{(n)}]$.
2) The convergence time is bounded above by $N_{\min} D$, and this convergence bound does not depend on the inconsistency in the situational awareness over the network.

IV. CBBA WITH DUO TASKS

Under the assumptions in section II-B, the score of two agents doing a duo task is expressed as the sum of scores of each agent doing its own single-equivalent tasks. Thus, it may look attractive to apply CBBA method to the duo tasking problem by treating the new decision variables defined in II-B as normal decision variables for single tasks, because then the resulting solution is expected to inherit the robust convergence properties of CBBA. However, this simple strategy might not ensure satisfaction of cooperation requirements. For example, an agent is not allowed to take both the leader and follower roles for the same PDTs as represented in (10), and the reward of doing a RDT is awarded only when two agents, each of which comes from a different set, are simultaneously assigned to the task, as described by the bilinear term in (13). Thus, modifications are required in the algorithms to ensure these constraints are satisfied. This section presents the required modifications to handle PDTs and RDTs.

A. Preferred Cooperation

1) Subtask Decomposition and Bidding Mechanism: With the additive reformulation, a PDT j is decomposed into two subtasks, $\bar{j}$ and $\tilde{j}$. The former indicates the leader portion of task j, and the latter denotes the follower portion of the task, and equivalent single-task scores are also defined such that:

$$c_{i\bar{j}} = \bar{c}_{ij}^p,\ c_{i\tilde{j}} = \tilde{c}_{ij}^p. \tag{15}$$

If simply applying the baseline CBBA with these new tasks $\bar{j}$ and $\tilde{j}$ with associated scores, it could be that both the leader and the follower subtasks are assigned to a single agent. In the CBBA framework that aims for agreement on the winning scores and winning agents, it is natural to encode mission constraints as appropriate scoring functions. In this sense, this paper proposes the following score modification

to avoid two subtasks decomposed from the same PDT being assigned to as single agent:

$$c'_{i\bar{j}} = c_{i\bar{j}} \times \mathbb{I}(\tilde{j} \notin \mathbf{b}_i) = c_{i\bar{j}} \times (1 - \mathbb{I}(z_{i\tilde{j}} = i)) \quad (16)$$

$$c'_{i\tilde{j}} = c_{i\tilde{j}} \times \mathbb{I}(\bar{j} \notin \mathbf{b}_i) = c_{i\tilde{j}} \times (1 - \mathbb{I}(z_{i\bar{j}} = i)). \quad (17)$$

In other words, agent i can place a nonzero bid on the leader subtask only when it is not assigned to the corresponding follower task, and vice versa. This forbids an agent adding both the leader and the follower tasks in its bundle. Note that the winning agent vector $\mathbf{z}_i$ changes over iterations. In the initial bundle construction, between the leader and the follower subtasks, an agent will choose the one that gives the greater score value. Once it selects one, the score of the other subtask becomes zero from (16) and (17), and the agent will not add the unselected subtask until it loses the chosen subtask. If the agent is outbid on the subtask it initially selected, the corresponding winning agent value changes and the other subtask now can provide non-zero score.

2) Convergence: As pointed out in section III, CBBA converges to a conflict-free assignment provided that every agent's scoring scheme satisfies the DMG property. Note that the score modification for the PDT subtasks described in (16) and (17) ensures that

$$c'_{i\bar{j}}[\mathbf{b}] \geq c'_{i\bar{j}}[\mathbf{b} \oplus_f \{\tilde{j}\}] = 0, \quad (18)$$

$$c'_{i\tilde{j}}[\mathbf{b}] \geq c'_{i\tilde{j}}[\mathbf{b} \oplus_f \{\bar{j}\}] = 0, \quad (19)$$

for any arbitrary bundle $\mathbf{b}$. Thus, once the original scoring schemes of each agent satisfy DMG, the task decomposition and the associated score modifications presented in this section lead to preserving the DMG property and the convergence properties.

B. Required Cooperation

1) Task Decomposition and Invalid Tasks: For a RDT j, two subtasks j_1 and j_2 are created to represent the type 1 and the type 2 portion of the duo task. Because each agent of one type is not compatible with the subtask of the other type, the following holds

$$c_{ij_2} = 0,\ c_{kj_1} = 0,\ \forall i \in \mathcal{I}_1,\ \forall k \in \mathcal{J}_2. \quad (20)$$

Before developing methods to address the required cooperation in the formulation in section II-A, first consider a special case where all the tasks are RDTs and therefore an agent of one type is only eligible for the corresponding set of subtasks. Define $\mathcal{J}_1$ and $\mathcal{J}_2$ as the sets of type 1 subtasks and type 2 subtasks, respectively; then the following holds:

$$c_{ij} = 0,\ c_{kl} = 0,\ \forall (i,j) \in \mathcal{I}_1 \times \mathcal{J}_2,\ \forall (k,l) \in \mathcal{I}_2 \times \mathcal{J}_1. \quad (21)$$

For this disjoint case, one natural decision framework is each group locally making their own plans based on the scores for the corresponding subtasks, i.e., $\hat{c}_{ij_1},\ i \in \mathcal{I}_1$ and $\hat{c}_{kj_2},\ k \in \mathcal{I}_2$ and then talking to the other team to create the coordinated solutions. After the local planning, for a certain RDT j, there could be three possible outcomes: (a) both subtasks j_1 and j_2 are assigned to some agent of type 1 and type 2, respectively; (b) neither of the subtasks are assigned; or, (c) only one subtask is assigned. For case (a), one can claim that the RDT j is assigned to an appropriate pair of agents, and for case (b), one can claim that the RDT j is not assigned because it is not attractive to the team. Case (c) can be problematic, because one agent is committed to take its part in RDT based on a wrong optimistic view, and actually obtains no reward from it. The performance degradation due to this *invalid* assignment can be larger than the loss of assumed score for the subtask the agent has committed to do. The reason for this additional performance degradation is the cost of opportunity. The agent may have been able to do other tasks had it been aware that this RDT would only be partially assigned, thus no actual reward could be obtained from it. Had the agent been aware of that fact, it would not have bid on the task in the first place. One approach to improve performance is to eliminate these invalid tasks and re-distribute the tasks accordingly.

Algorithm 1 Decentralized task elimination for agent i

```
Initialize invalid tasks set: J_IV = ∅.
Initialize outer-loop iteration count: t_out = 0.
while J_IV ≠ ∅ OR t_out < 1 do
    Eliminate invalid tasks: c_{ij1} = 0, c_{ij2} = 0, ∀j ∈ J_IV.
    t_out = t_out + 1.
    while CBBA not converged do
        CBBA bundle construction phase.
        CBBA conflict resolution phase.
        Check CBBA convergence.
    end while
    Identify invalid tasks: J_IV ≜ {j ∈ J_RD | (z_{ij1} = NULL) XOR (z_{ij2} = NULL)}
end while
```

To this end, this section presents methods that accomplish this task elimination and re-distribution in a *decentralized* manner, and finally create a conflict-free assignment without any invalid assignments.

2) Decentralized Task Elimination: Suppose that CBBA is run to convergence for the heterogeneous team $\mathcal{I}_1 \cup \mathcal{I}_2$ with defining subtasks and associated scores as (20) for required duo tasks in $\mathcal{J}_{RD}$ (and applying the techniques in section IV-B for RDTs). Then, each agent will have agreed on some conflict-free assignment treating the subtasks as single tasks. If there are no invalid tasks, it is a feasible assignment to the task allocation problem with RDTs. If there are invalid tasks in this first set of solution, it is desirable to re-assign tasks so that there are no invalid tasks.

If every agent's scoring schemes satisfy DMG, and thus CBBA has converged to a conflict-free solution, the following facts hold for the converged solution:

1) Every agent has the same winning agents vector, $\mathbf{z}_i,\ i \in \mathcal{I}$, and winning bids vector, $\mathbf{y}_i,\ i \in \mathcal{I}$.
2) Every agent can identify all the invalid tasks only using the local information $\mathbf{z}_i$.
3) For two tasks j and k that are assigned to the same agent l, i.e., two tasks and the associated agent satisfying $z_{ij} = z_{ik} = l$, $y_{ij} > y_{ik}$ if and only if task j is

located in the earlier position in agent l's bundle than task k.

4) Any agent $i \in \mathcal{I}$ can re-construct every other agent's bundle from only $\mathbf{z}_i$ and $\mathbf{y}_i$.

Thus, once CBBA has converged, every agent has the same knowledge about the status of RDTs, i.e., whether or not they are validly assigned. With this agreed-on information, one can create a more desirable solution by excluding the invalid tasks out of the task list and then running another round of CBBA. The process of repeating task elimination and CBBA execution will eventually create a feasible solution without any invalid assignment because there are a finite number of RDTs. This is the basic concept of the decentralized task elimination (DTE) in this work, and Algorithm 1 summarizes the process for an arbitrary agent i. Identification of invalid tasks in line 11 and elimination of them in line 4 rely only on the up-to-date local information $\mathbf{z}_i$ and $\mathbf{y}_i$, thus are performed in a decentralized way. The while loop starting from line IV-B.2 represents a single run of CBBA to convergence. Note that over a strongly connected network, the convergence check of CBBA in line 9 can also be done in a decentralized manner: CBBA has converged if $\mathbf{z}_i$ and $\mathbf{y}_i$ has been the same for D communication rounds where $D < N_a$ is the network diameter. Thus, over a strongly connected network, the only part of the DTE procedure that requires communication with other agents is the CBBA conflict resolution process. Notice that the CBBA loop from line converges more quickly in the later outer iteration, i.e., larger t_{out}, than the earlier iteration. This is because elimination of invalid tasks does not affect the assignments that were more valuable than the largest-scored invalid task.

C. Fixed-Time Waiting

As explained, the DTE method guarantees convergence to a conflict-free solution for required duo cooperation. However, since it runs CBBA multiple times, its run-time could be relatively long compared to a single run of CBBA. Extensive numerical experiments have suggested that many assignment are locked in in the earlier phase of the CBBA process; this leads to suggestion of an alternative way of performing required duo assignment that improves the run-time efficiency. The way is to allow agents to make decisions about the validity of duo assignment before full CBBA convergence. Initially, agent $i \in \mathcal{I}_1$ is allowed to optimistically place a bid on the corresponding portion of some RDT. But, if it does not find an agent $k \in \mathcal{I}_2$ that places a bid on the other portion of the RDT within some waiting time Δ_{wait}, agent i drops the subtask and disables bidding on that particular task. Later, it can enable the bidding only when it finds a partner agent bidding on that subtask. Note that all information needed for these switching decisions can be extracted from the winning bids list and the winning agents list. Under the same set of assumptions for the CBBA convergence, this switching protocol does not guarantee convergence to a conflict-free solution, and the solution could be cycling. However, for most physically-meaningful scenarios, conflict-free solutions are obtained, as demonstarted through simulation. Moreover, this method is convenient for incorporating the temporal cooperation constraints of the two agents performing the duo task – e.g., the two agents should arrive at the duo task with overlapping period of time.

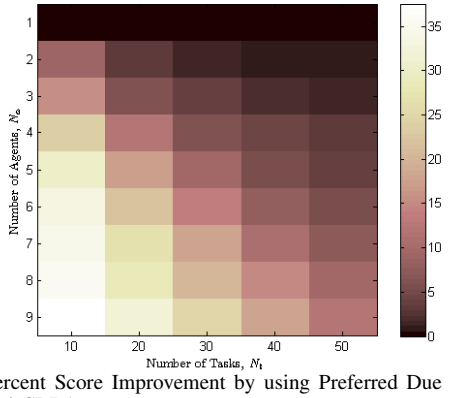

Fig. 1. Percent Score Improvement by using Preferred Due cooperation over Original CBBA

V. Numerical Results

A. Simulation Environment

The performance of the algorithms presented here are analyzed by running Monte-Carlo simulations for a general scenario. In this scenario, there are N_t tasks to be completed by N_a agents. The two task types are Tracking, $\mathcal{J}_{track}$ and Striking, $\mathcal{J}_{strike}$. A track task requires one agent with tracking capability if it is a ST, and can accommodate two agents with tracking capability if it is a PDT. In this scenario, rules of engagement dictate that the strike task must have one agent with strike capability for weapons release, and a separate agent with tracking capability to spot the target as it is engaged. There are 2 agent types: agents in $\mathcal{I}_1$ are sensor UAVs with tracking capability, and agents in $\mathcal{I}_2$ are WUAVs with strike capability. The agents of type 2 may or may not have tracking capability as well. At the beginning of each simulation the agent and task locations are randomly initialized in a continuous 200 by 200 world. The maximum number of possible tasks assigned is constrained by the fact that tasks have specified time windows associated with them. Only tasks that are begun within the allowed time windows give the agent non-zero scores. The start of the time windows in these simulations are chosen randomly.

B. Performance Improvement of Preferred Duo cooperation

This section demonstrates the performance improvement by the preferred duo cooperation capability. The baseline CBBA, which treats every task as a solo task, provides the benchmark against the assignment from the preferred cooperation method. In this simulation, $N_a \in \{1, \ldots, 9\}$ and $N_t \in \{10, 20, \ldots, 50\}$ are used. All tasks are of type $\mathcal{J}_{track}$, and each case was run 100 times. In the baseline CBBA, $c_{ij} = 100$, and at most one agent can be assigned per task. In the preferred duo cooperation up to two agents can be assigned to each task. The scores for the leader role and the follower role are $c_{i\bar{j}} = 100e^{-\lambda\tau_{ij}}$ and $c_{i\tilde{j}} = 50e^{-\lambda\tau_{ij}}$, respectively, where τ_{ij} is the travel time of agent i to task j and λ is the discount factor penalizing late arrival. Note

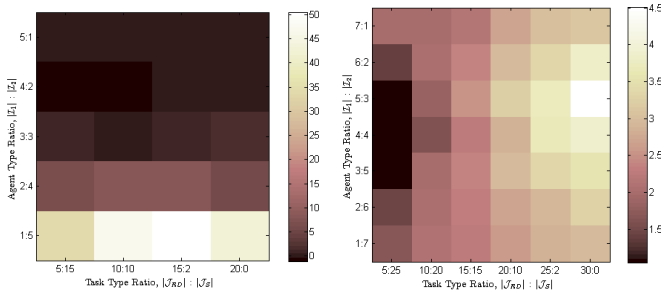

Fig. 2. Performance and runtime comparison for DTE

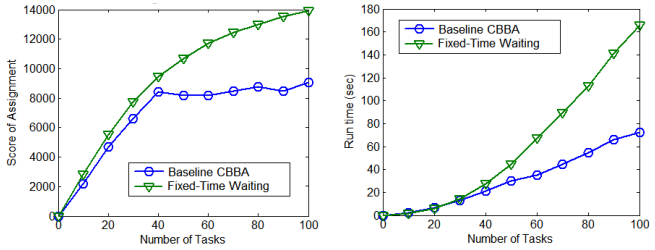

Fig. 3. Performance and runtime comparison for fixed-time waiting

that with this setting where the follower score is a half of the leader score, the maximum potential performance improvement by the preferred cooperation is order of 50%.

Fig. 1 shows the improvement in the total score of the assignment (sum of the scores of the tasks assigned) due to the capability of handling duo tasks. Notice that the benefit is most prominent when the $N_t = 10$ and $N_a = 9$, so that the ratio $N_t/N_a \approx 1$, providing over 35% performance benefit. This is expected, because for such cases a high percentage of the agents are available (unassigned) at any given time. Therefore, allowing these unassigned agents to cooperate on a task can give substantial improvement to the overall score. In the case where N_t/N_a is high, the preferred duo cooperation does not adversely affect the overall score, but the improvement is small. The relatively small improvement is due to the fact that, given many tasks to choose from, agents typically select the higher utility leader task over the lower utility follower task.

C. Performance Improvement for Required cooperation

First, the performance and runtime of the DTE scheme are compared with those of the local task elimination (LTE) that simply deletes the invalid assignments from the execution list after running the baseline CBBA for a single time. The ratio of WUAVs to SUAV is varied as a parameter, as is the ratio of joint (strike) tasks to solo (track) tasks. It can be seen in Fig. 2 that DTE improves the performance up to 17% at the cost of runtime increase by the factor of 2-3. Note that this amount of increment in runtime is typically still acceptable, as the baseline CBBA converges very quickly.

Second, the performance and runtime of the fixed-time waiting scheme are tested with $\Delta_{wait} = 2D$ where D is the network diameter. The number of agents is 20 and the number of joint tasks is varied from 10 to 100. Fig. 3 demonstrates that the waiting scheme provides significant performance improvement for large tasking problems.

VI. Conclusions

This paper extended the consensus-based bundle algorithm to account for cooperation requirements and preference often defined by rules of engagement in complex multi-agent missions. Task decompositions and associated scoring modifications were developed to allow for soft-constrained cooperation preference, and decentralized task elimination protocols were proposed to ensure satisfaction of hard-constrained cooperation requirements. Numerical results with cooperation of sensor UAVs and weaponized UAVs were presented to verify the proposed algorithms. Future work will address more specified temporal cooperation requirements.

Acknowledgments

This work is funded in part by AFOSR # FA9550-08-1-0086, and by ONR STTR # N00014-08-C-0707 (with Mr. Olivier Toupet at Aurora Flight Sciences). The authors would like to thank Cameron Fraser and Sameera Ponda for inspiring discussions.

References

[1] J. Bellingham, M. Tillerson, A. Richards, and J. How, "Multi-task allocation and path planning for cooperating UAVs," in *Proceedings of Conference of Cooperative Control and Optimization*, Nov. 2001.

[2] C. Schumacher, P. Chandler, and S. Rasmussen, "Task allocation for wide area search munitions," in *Proceedings of the American Control Conference*, 2002.

[3] C. Cassandras and W. Li, "A receding horizon approach for solving some cooperative control problems," in *Proceedings of the IEEE Conference on Decision and Control*, 2002.

[4] Y. Jin, A. Minai, and M. Polycarpou, "Cooperative real-time search and task allocation in UAV teams," in *Proceedings of the IEEE Conference on Decision and Control*, 2003.

[5] D. Turra, L. Pollini, and M. Innocenti, "Fast unmanned vehicles task allocation with moving targets," in *Proceedings of the IEEE Conference on Decision and Control*, Dec 2004.

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

[7] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative-timing missions," *Journal of Guidance, Control, and Dynamics*, vol. 28(1), pp. 150–161, 2005.

[8] D. Castanon and C. Wu, "Distributed algorithms for dynamic reassignment," in *Proceedings of the IEEE Conference of Decision and Control*, 2003.

[9] J. Curtis and R. Murphey, "Simultaneous area search and task assignment for a team of cooperative agents," in *Proceedings of AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003.

[10] T. Shima, S. J. Rasmussen, and P. Chandler, "UAV team decision and control using efficient collaborative estimation," in *Proceedings of the American Control Conference*, 2005.

[11] M. Alighanbari, "Robust distributed task assignment algorithms," Ph.D. dissertation, Department of Aeronautics and Astronautics, MIT, August, 2007.

[12] M. Alighanbari and J. P. How, "A Robust Approach to the UAV Task Assignment Problem," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 2, pp. 118–134, Jan 2008.

[13] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. on Robotics*, vol. 25 (4), pp. 912 – 926, 2009.

[14] S. Ponda, J. Redding, H.-L. Choi, B. Bethke, J. P. How, M. Vavrina, and J. L. Vian, "Decentralized planning for complex missions with dynamic communication constraints," in *submitted to American Control Conference*, 2009.

[15] L. Bertuccelli, H.-L. Choi, , P. Cho, and J. P. How, "Real-time multi-UAV task assignment in dynamic and uncertain environments," in *Proceedings of AIAA Guidance, Navigation and Control Conference*, 2009.