2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Communication-Constrained Distributed Task Assignment

Justin Jackson*, Mariam Faied†, Pierre Kabamba‡, Anouck Girard§
*PhD Candidate, Aerospace Engineering, Email: jpjack@umich.edu
†Postdoctoral Researcher, Aerospace Engineering
‡Professor, Aerospace Engineering
§Assistant Professor, Aerospace Engineering,
University of Michigan, Ann Arbor, Michigan 48109

Abstract— **This paper considers the problem of distributed assignment of tasks to agents in the presence of task constraints, where the agents use a known, but arbitrary communication topology. The task assignment problem considered here requires that all agents that perform tasks related by a task constraint be able to communicate directly. The problem is motivated by complex military missions where tasks are assigned to various vehicles and tasks must be scheduled to meet constraints between them. This requires communication between vehicles responsible for tasks that are related by constraints. The physically distributed and dynamic nature of such missions combined with unreliable communication motivates algorithms that can perform the required distributed planning. Toward this goal, we introduce a method that assigns tasks under the restrictions imposed by these mission constraints. The new method presented here is a distributed search designed to solve a nonlinear, distributed constrained assignment problem for which a proof of correctness is presented. The method is illustrated on an example involving two unmanned air vehicles and two unmanned ground vehicles.**

I. INTRODUCTION

Consider the following motivational example. Two unmanned air vehicles (UAVs) and two unmanned ground vehicles (UGVs) are tasked to prosecute two targets. Each vehicle has only local knowledge of the structure of the communication network. In this scenario, the communication topology is limited as depicted in Figure 1. Actual causes of such situations may include range limitations, terrain, and heterogeneous communication protocols. These limitations motivate the development of planning algorithms that can operate in the presence of a arbitrary communication networks.

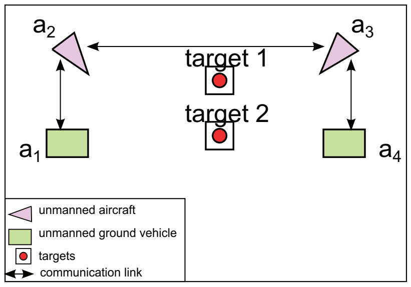

Fig. 1. UAV and UGV example.

The example mission in Figure 1 requires a plan where both targets are prosecuted. To achieve this, a UGV must track a target while a UAV attacks it. The tracking vehicle must be in communication with the attacking vehicle so that the attack can be coordinated. The vehicles must use information related to vehicle capability, task constraints, and the locally known communication topology to decide which vehicles will perform each of these tasks on each of the targets while obeying the given restrictions. This is an example of a distributed system, i.e., a system with one or both of the following properties: 1) no agent has complete knowledge of the full system, 2) no agent has complete control over the actions of the other agents. This motivating example was chosen because it features vehicles with heterogeneous abilities, a connected communication topology, and several heterogeneous tasks. In spite of its small size, it contains all of the necessary ingredients to illustrate the concepts of this paper. While this example is military in nature, our notions of tasks, agents, communication, and clustering constraints are general and are applicable to numerous other constrained assignment applications such as those in [1], [2].

This paper introduces the Communication-Constrained Distributed Assignment Problem (CDAP). The CDAP concerns assigning tasks that are related by constraints, to agents such that those assigned related tasks have the capability to communicate. This ability to communicate ensures that the constraints relating these tasks can be satisfied if possible. The CDAP is a nonlinear distributed constrained assignment problem that we solve using a stochastic bidding method. The effect of this bidding method is that the global exchange of tasks is done in a way reminiscent of optimization by Simulated Annealing. The computation associated with optimization by Simulated Annealing can be distributed (or parallelized) [3], but our method distributes the information associated with the representation of the solution. To the authors' knowledge, this aspect is previously unexplored.

A. Literature Review

Three core motivating problems in the scheduling and planning literature are the Assignment Problem [4], the Job-Shop Scheduling Problem (JSP) [5], and the Traveling Salesman Problem (TSP) [6]. Problems involving vehicle routing, assignment, and scheduling necessarily contain elements of all three. The TSP formulation has become a widely used tool for solving vehicle routing problems (VRP) and many other combinatorial problems [7]. The JSP concerns the scheduling of several jobs on a number of machines such that the lateness of job completion is minimized [8].

Military mission planning problems are often formulated as scheduling and assignment problems [9], [10]. Assignment and scheduling problems have originated in operations research where a large literature exists [8]. Modern practical planning problems have motivated an expansion of this literature into diverse areas including electrical engineering [11], computer science [12], and aerospace engineering [13].

Distributed systems theory is often concerned with problems of efficient communication over networks, consensus regarding data access and management, and fault-tolerance or resilience to failure [14], [15]. Solution methods for distributed constraint satisfaction problems (DCSP) were pioneered by Yokoo [16] and include distributed backtracking and asynchronous weak commitment search. These methods can be used to solve problems involving setting the values of several variables that are related by predicate constraints.

Auction methods are a reliable, low-complexity way to find near-optimal [13] and in some cases, optimal solutions [17] to assignment problems. Distributed assignment methods for UAVs include Capacitated Transhipment Assignment formulations [18], which perform assignment in a greedy way, but are effective and can be done in real-time. The problem of information consensus becomes important to distributed estimation and information fusion problems [19]. Consensus methods can be useful for determining continuous quantities like the positions and velocities of vehicles [20]. Sophisticated methods such as distributed integer linear programming (ILP) methods [21] have been designed to solve linear, multi-vehicle assignment problems in the presence of communication delays. In spite of these extensive tools, there has been no previous work in distributed systems designed to solve the nonlinear distributed constrained assignment problem developed in this paper.

B. Original Contributions

The primary contributions of this work are as follows.

1) An introduction to the Communication-Constrained Distributed Assignment Problem (CDAP) is given.
2) A distributed algorithm that solves this problem is presented and its correctness is proven.

In contrast to other task assignment formulations, including the methods discussed above, the CDAP incorporates the constraints that require that those agents which are assigned to tasks related by constraints be able to communicate with each other. We also develop a unique formulation that converts the distributed constrained assignment problem into a distributed optimization problem. A new Stochastic Bidding Algorithm (SBA) is designed and used to solve the resulting distributed optimization problem. Unlike other auction methods, the SBA effectively incorporates randomness to find a global minimum of the objective function. This method requires only local information about the communication network topology. The SBA terminates if and only if a solution to the CDAP is found. The SBA has probabilistic completeness properties, but this is beyond the scope of the current paper. The novelty of this approach, as compared to others in the literature, is to explicitly address the issues of distributed data and authority in a communication-constrained assignment problem. Other methods either solve the problem in a centralized way [1], [2], [17], do not address communication-constrained assignment [17], [13], [22], or require that all agents be able to communicate directly [22]. All of these issues are addressed in the current paper.

II. Overview

This paper is organized as follows: Section III develops the notion and conventions used throughout; Section IV details the Communication-Constrained Distributed Assignment Problem; Section V illustrates the approach used to solve this problem; Section VI details the Stochastic Bidding Algorithm; Section VII presents an exposition of the execution of the SBA and demonstrates its effectiveness; and Section VIII concludes the paper.

III. Notation and Preliminaries

This section details the concepts that are used to formulate the problem of this paper. The set of tasks to be assigned is

$$\mathcal{T} = \{t_1, \ldots, t_{N_t}\}, \tag{1}$$

where $N_t > 0$ is the number of tasks. In the motivating example $N_t = 4$ and $t_1 \equiv$ track target 1; $t_2 \equiv$ attack target 1; $t_3 \equiv$ track target 2; and $t_4 \equiv$ attack target 2. The set of tasks is therefore $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$.

Tasks are to be assigned to agents. The set of agents is

$$\mathcal{A} = \{a_1, \ldots, a_{N_a}\}, \tag{2}$$

where $N_a > 0$ is the number of agents. The agents in the motivational example are $a_1 \equiv$ left UGV; $a_2 \equiv$ left UAV; $a_3 \equiv$ right UAV; and $a_4 \equiv$ right UAV. The set of agents is therefore $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$.

A *task assignment* is a mapping from tasks to agents. It is defined formally as,

$$TA : \mathcal{T} \to \mathcal{A}, \tag{3}$$

and tells which agents will perform each of the tasks. A task assignment is a mapping (as opposed to simply a relation). Hence, we require that each task be assigned to one and only one agent. This mapping need not be injective, i.e., an agent may be assigned several tasks. This mapping need not be surjective, i.e., an agent may not be assigned any task at all. Note that there is no loss of generality in requiring that a task assignment be a mapping: if a task requires several agents, it should be split into subtasks requiring one agent each. An example task assignment for the motivational example is

$$TA_1 = \{(t_1, a_4), (t_2, a_2), (t_3, a_4), (t_4, a_3)\}. \tag{4}$$

The first notion of feasibility used in this paper is feasibility with respect to capability and is described using a relation from $\mathcal{T}$ to $\mathcal{A}$, i.e., a subset of their Cartesian product:

$$Capability \subseteq \mathcal{T} \times \mathcal{A}. \tag{5}$$

A pair $(t, a) \in \mathcal{T} \times \mathcal{A}$ is in *Capability* iff task t *can be performed by* agent a. This describes the physical capability of the agents to perform the tasks. Without loss of generality, we assume that the range of Capability is $\mathcal{A}$. In other words, we assume that each agent is capable of performing at least one task. The relation *Capability* for the motivational example is

$$Capability = \{(t_1, a_1),(t_1, a_4),(t_2, a_2),(t_2, a_3), (t_3, a_1),(t_3, a_4),(t_4, a_2),(t_4, a_3)\}. \tag{6}$$

A task assignment is called *feasible with respect to capability* iff $TA \subseteq Capability$, i.e.,

$$(t_i, TA(t_i)) \in Capability, i = 1, \ldots, N_t. \tag{7}$$

In practice, this means that each task is assigned to an agent that is capable of performing it. For the motivating example, the task assignment of (4) is feasible with respect to capability whereas the task assignment,

$$TA_2 = \{(t_1, a_1),(t_2, a_1),(t_3, a_4),(t_4, a_4)\}, \tag{8}$$

is not. The task assignment TA_2 is not feasible with respect to capability because tasks t_2 and t_4, which are attacking tasks, are not assigned to UAVs that are capable of performing them.

A central idea of this paper is that tasks are bound to each other in the following sense. Tasks may be related by operational constraints and the agents that are assigned such related tasks must be able to communicate in order to properly plan for and perform these tasks. In the motivating example, tracking and attacking tasks for a single target are related by the operational constraint that they must be performed at the same time (i.e., when scheduled, these tasks must be scheduled to occur at the same time). The scheduling of these tasks is left for future work. The effect of the constraints that bind tasks is described formally as follows, consider $N_{cl} > 0$ task clusters,

$$\mathcal{T}_1, \ldots, \mathcal{T}_{N_{cl}} \subseteq \mathcal{T}, \tag{9}$$

each of which represents a particular constraint and contains as elements, the tasks that are involved in each such constraint. The task clusters for the motivating example are $\mathcal{T}_1 = \{t_1, t_2\}$ and $\mathcal{T}_2 = \{t_3, t_4\}$.

The concept of a cluster union is also used in this work. Informally, a cluster union is defined for each task t_i and is the set of tasks with which task t_i shares a cluster. Formally a cluster union is,

$$\mathcal{C}_i = \{t_k \mid \exists m \leq N_{cl} : t_i \in \mathcal{T}_m \text{ and } t_k \in \mathcal{T}_m\}, i = 1, \ldots, N_t. \tag{10}$$

The cluster unions for each of the tasks in the motivating example are, $\mathcal{C}_1 = \{t_2\}$; $\mathcal{C}_2 = \{t_1\}$; $\mathcal{C}_3 = \{t_4\}$; and $\mathcal{C}_4 = \{t_3\}$.

We use the following standard notions from graph theory. An (undirected) *graph* is a pair $(\mathcal{V}, \mathcal{E})$ of vertices and edges such that each edge is a couple of vertices [23]. A graph is called *complete* iff every couple of vertices is an edge. For the graph $(\mathcal{V}, \mathcal{E})$, if $\mathcal{V}' \subseteq \mathcal{V}$, the *subgraph induced by restriction to* $\mathcal{V}'$, denoted $(\mathcal{V}, \mathcal{E}) \mid_{\mathcal{V}'}$, is the graph $(\mathcal{V}', \mathcal{E}')$, where

$$\mathcal{E}' = \{\{v_1, v_2\} \in \mathcal{E} \mid v_1 \in \mathcal{V}' \text{ and } v_2 \in \mathcal{V}'\}. \tag{11}$$

In other words, the induced subgraph is obtained by retaining only vertices in $\mathcal{V}'$ and the edges connecting them. The *distance* between two vertices $v, w \in \mathcal{V}$ is $d(v, w)$ and represents the number of edges that must be traversed to move from v to w across the graph. The *diameter* of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is,

$$diam(\mathcal{G}) = \max_{v,w \in \mathcal{V}} d(v, w). \tag{12}$$

The *neighborhood* of a vertex $v \in \mathcal{V}$ is the set $\mathcal{N}_v = \{w \in \mathcal{V} \mid e = \{v, w\} \in \mathcal{E}\}$.

The agents in (2) have communication capability described by an undirected, connected communication graph,

$$\mathcal{G}_c = (\mathcal{A}, \mathcal{E}_c). \tag{13}$$

There is an edge between two agents iff they are able to communicate directly with each other. The type of communication assumed here is acknowledgement-based, where each agent knows when communication is established with another agent. The edge set of the communication graph for the motivating example is,

$$\mathcal{E}_c = \{\{a_1, a_2\}, \{a_2, a_3\}, \{a_3, a_4\}\}. \tag{14}$$

A task assignment is said to be *feasible with respect to clustering* iff

$$(\mathcal{A}, \mathcal{E}_c) \mid_{TA(\mathcal{T}_i)} \text{ is complete}, i = 1, \ldots, N_{cl}. \tag{15}$$

Requirement (15) means that the agents that are assigned to the tasks belonging to a cluster must all be able to communicate directly with each other. The task assignment of (8) is feasible with respect to clustering, whereas the task assignment of (4) is not. The infeasibility of the task assignment TA_1 with respect to clustering results because agent $TA(t_1)$ cannot directly communicate with agent $TA(t_2)$ although t_1 and t_2 belong to the same cluster $\mathcal{T}_1$.

Definition Feasible task assignment:
A task assignment that is feasible with respect to capability and feasible with respect to task clustering is said to be a *feasible* task assignment.

For the motivational example, the task assignment,

$$TA_3 = \{(t_1, a_1),(t_2, a_2),(t_3, a_1),(t_4, a_2)\}, \tag{16}$$

is feasible.

IV. Problem Definition

Each agent a_j is assumed to have the know following data:

1) The tasks t_i such that $(t_i, a_j) \in Capability$,
2) For all such t_i, all clusters $\mathcal{T}_m$ such that $t_i \in \mathcal{T}_m$,
3) Its neighborhood $\mathcal{N}_{a_j}$ on the communication graph,
4) The tasks t_l such that $(t_l, a_k) \in Capability$ and $a_k \in \mathcal{N}_{a_j}$,

where $i, l = 1, \ldots, N_t$; $j, k = 1, \ldots, N_a$; and $m = 1, \ldots, N_{cl}$. The problem discussed here is for the agents in $\mathcal{A}$, to collectively find a feasible task assignment TA, using only the available data together with communication with neighbors by (13). This is the Communication-Constrained Distributed Assignment Problem (CDAP). Propositions 4.1 and 4.2 help to underscore the difficulty of solving the CDAP. Proofs of propositions are in the appendix.

Proposition 4.1: The number of possible task assignments grows as $\mathcal{O}(N_a^{N_t})$.

For the motivating example there are $4^4 = 256$ possible assignments.

Proposition 4.2: Let $X \in \{0,1\}^{N_t \times N_a}$ represent a task assignment so that $X_{ij} = 1$ if $TA(t_i) = a_j$ and $X_{ij} = 0$ otherwise. Then the CDAP problem can be formulated as a system of nonlinear equations in X.

The difficulty of the problem is highlighted by Propositions 4.1 and 4.2 and the fact that the input data and computational resources are distributed across a communication network of arbitrary topology. The size of the problem is polynomial in the number of agents and exponential in the number of tasks. Exhaustive enumeration is infeasible due to complexity and the distributed nature of the problem data.

V. Technical Approach

This section describes the approach used to solve the CDAP problem and further develops the tools used in this approach. Define a set of messages $\mathcal{M}$, possibly infinite and closed under union. The content that the agents communicate to their neighbors originates in this set of messages. That specific content is discussed later.

For every multi-index (i, j) such that $(t_i, a_j) \in Capability$, define a quadruple,

$$[t_i, a_j] = (States_{ij}, start_{ij}, trans_{ij}, msgs_{ij}). \tag{17}$$

This quadruple is called a *process*, where $States_{ij}$ is the state space of process $[t_i, a_j]$, i.e., a set of configuration quantities that may be boolean, integer, or real valued that describe the configuration of the process and represent its memory; $start_{ij} \in States_{ij}$ is the state at which process $[t_i, a_j]$ begins operation;

$$\begin{aligned} trans_{ij} &: \mathcal{M} \times States_{ij} \to States_{ij}, && (18) \\ msgs_{ij} &: \mathcal{M} \times States_{ij} \to \mathcal{M}. && (19) \end{aligned}$$

Processes advance this state appropriately through the function $trans_{ij}$, which accepts incoming messages and produces a new state from the current state. The function $msgs_{ij}$ is responsible for reading received messages, the new state, and based on these, sending appropriate messages. Let $Processes$ be the set of processes defined in (17).

Define the undirected process graph $\mathcal{G}_p = (Processes, \mathcal{E}_p)$, where

$$\mathcal{E}_p = \{\{[t_i, a_j], [t_k, a_l]\} \mid \{a_j, a_l\} \in \mathcal{E}_c\}. \tag{20}$$

The process $[t_i, a_j]$ is connected to the process $[t_k, a_l]$ if and only if agent a_j is connected to agent a_l by a communication link. The process graph for the motivating example is shown in Figure 2.

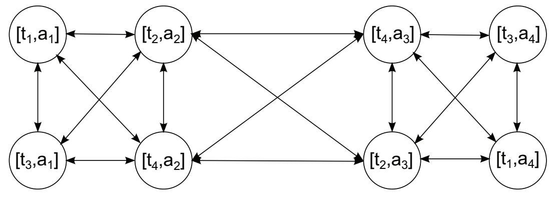

Fig. 2. Process graph for the example of Section I.

The vertex set of the process graph for the motivating example is given by (6) and the edge set follows from (20) and (14).

The processes $[t_i, a_j] \in Processes$, form a distributed system in the sense specified in Section I. This paper considers this distributed system under synchronous operation. That is, the processes each simultaneously update their state and then simultaneously send messages to their neighbors. Each iteration of computation and message transmission is referred to as a *round*.

The algorithm developed in this work allows processes to bid on behalf of their agents for the tasks that the corresponding agent is capable of performing. Tasks are bid on in numerical round-robin order, when the bidding for one task is finished, bidding for the next one begins. Algorithms 1 and 2 detail the operation of the $trans_{ij}$ and $msgs_{ij}$ functions respectively for the algorithm presented in this paper. The bidding procedure is depicted graphically in Figure 3. Note that the diagram in Figure 3 does not terminate. Distributed algorithms often have an associated termination condition that stops the algorithm from executing when a solution is found [14]. This termination condition is usually a function of the states of each process and thus requires current knowledge of each process which in general, no process will have. Rather than terminate, it is enough that there exists a round at which the processes collectively output a solution.

The idea of an assignable process informally means that if a process has won the bidding for its task and its neighbors have won theirs, that process will then satisfy all clustering constraints associated with its task. Formally an assignable process is defined as follows,

Definition Assignable Process:
process $[t_i, a_j]$ is assignable if $\forall t_k \in \mathcal{C}_i$, $\exists [t_k, a_l] \in \mathcal{N}_{[i,j]}$.

A process is unassignable if it is not assignable. Unassignable processes are not able to satisfy clustering constraints, these processes do not participate in the bidding

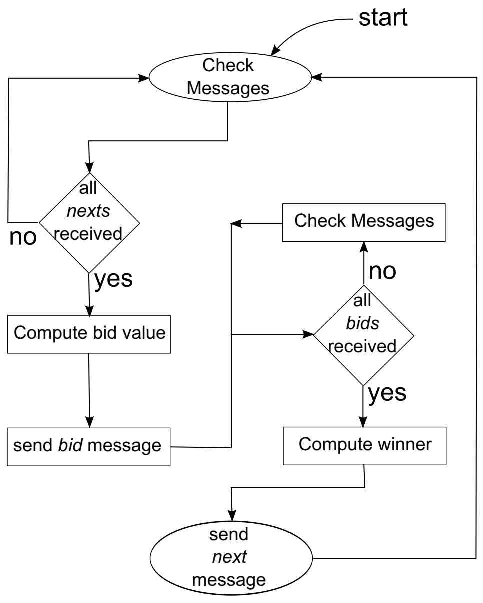

Fig. 3. Bidding procedure diagram.

procedure. To illustrate this concept, consider the modification of the example of Section I obtained by disabling the right UGV (i.e. agent a_4) as shown in Figure 4.

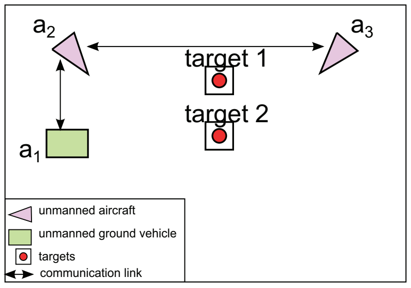

Fig. 4. Modified UAV and UGV example.

The set of tasks remains the same, but the new set of agents and the relation $Capability$ are,

$$\mathcal{A} = \{a_1, a_2, a_3\}, \tag{21}$$

and

$$Capability = \{(t_1, a_1), (t_2, a_2), (t_2, a_3), (t_3, a_1), (t_4, a_2), (t_4, a_3)\}. \tag{22}$$

The resulting process graph is shown in Figure 5.

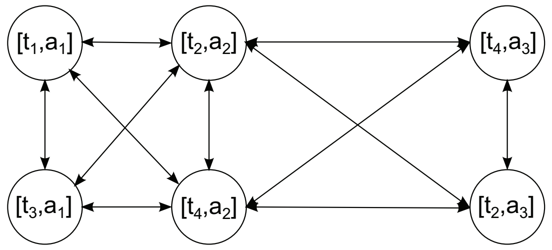

Fig. 5. Process graph for the modified example.

Notice that in Figure 5, the processes $[t_4, a_3]$ and $[t_2, a_3]$ do not share edges with any processes $[t_3, a_j]$ or $[t_1, a_l]$ respectively. This results in processes $[t_4, a_3]$ and $[t_2, a_3]$ being unable to satisfy clustering constraints, $\mathcal{T}_2$ and $\mathcal{T}_1$ respectively. These processes are therefore *unassignable*.

VI. Solution Procedure

The remainder of this paper details the bidding method used for solving the Communication-Constrained Distributed Assignment Problem. The approach is to restructure the CDAP as a minimization problem, employ tools from distributed auctions to find minima of the resulting cost function, and use stochastic bidding to ensure a global minimum is found. The uniqueness of the assignment for each task is satisfied by a conflict resolution method [13].

Let TA be the current task assignment across the system, not necessarily feasible with respect to clustering. Define for each process $[t_i, a_j]$, a quantity that specifies whether or not it has won the bidding for its task. This quantity, introduced in Proposition 4.2, is

$$X_{ij} = \begin{cases} 1 & \text{if } TA(t_i) = a_j \\ 0 & \text{otherwise} \end{cases} . \tag{23}$$

For the motivational example, consider process $[t_1, a_1]$, process $[t_1, a_4]$, and the task assignment of (4). Here, $X_{11} = 0$ and $X_{14} = 1$.

The set of neighboring processes that allow process $[t_i, a_j]$ to satisfy the clustering constraints associated with task t_i is defined as,

$$\mathcal{NC}_{ij} = \{[k, l] \in \mathcal{N}_{[i,j]} \mid X_{kl} = 1 \text{ and } t_k \in \mathcal{C}_i\}, \tag{24}$$

and the cardinality of this set is,

$$nc_{ij} = \mid \mathcal{NC}_{ij} \mid . \tag{25}$$

If $\mid \mathcal{NC}_{ij} \mid = \mid \mathcal{C}_i \mid$ and $X_{ij} = 1$, then all required clustering constraints for task t_i are satisfied. For process $[t_1, a_1]$ and process $[t_1, a_4]$ and the task assignment of (4), $\mathcal{NC}_{11} = \{[t_2, a_2]\}$ and $\mathcal{NC}_{14} = \emptyset$. Hence, $nc_{11} = 1$ and $nc_{14} = 0$.

Consider the set of all assignable processes that bid on a task t_i. This set is

$$\mathcal{B}_i = \{[t_i, a_j] \in Processes \mid [t_i, a_j] \text{ is assignable}\}, \quad i = 1, \ldots, N_t. \tag{26}$$

For the example of Section I and task t_1, $\mathcal{B}_1 = \{[t_1, a_1], [t_1, a_4]\}$.

The *deficiency* of a process (with respect to the connections it must make as required by clustering) is defined as:

$$nd_{ij} = \mid \mathcal{C}_i \mid - nc_{ij}. \tag{27}$$

The sum of this deficiency across the process graph is,

$$J(TA) = \sum_{i,j:[t_i,a_j] \in Processes} nd_{ij} \cdot X_{ij}. \tag{28}$$

Define the optimization problem,

$$\min_{TA \in \mathcal{A}^{\mathcal{T}}} J(TA) \tag{29}$$

$$\text{s.t. } TA \subseteq Capability.$$

The minimum value of any process deficiency, nd_{ij} is zero and corresponds to $TA(t_i)$ being in communication with all agents $TA(t_k)$ where $t_k \in \mathcal{C}_i$, $i = 1, \ldots, N_t$. Proposition 6.1 states the equivalence of solving this optimization problem and the solution of the CDAP. The proof of this proposition is in the appendix.

Proposition 6.1: $J(TA) = 0$ if and only if TA satisfies (15).

Note that nd_{ij} is a function of X_{kl}, where $[t_k, a_l] \in \mathcal{N}_{[t_i,a_j]}$, this introduces a nonlinearity into the cost function (29). We do not expect $J(TA)$ to have unique minimizer, i.e., feasible assignments may not be unique.

A. Stochastic Bidding Algorithm

The bidding algorithm presented below attempts to minimize the cost function in (29) and thus find a task assignment that satisfies (15). This bidding algorithm favors processes that satisfy their respective clustering requirements dictated by $\mathcal{T}_m, m = 1, \ldots, N_{cl}$. The SBA assumes, without loss of generality, that all tasks are bid on in numerical round-robin order and is informally described as follows. The bidding begins with task t_1. Bidding begins for $t_i \in \mathcal{T}$ when a process $[t_i, a_j] \in \mathcal{B}_i$, has received $next$ messages from every $[t_{(i-1)}, a_j] \in \mathcal{B}_{i-1}$, where t_0 is defined as t_{N_t} by round-robin. When a process bidding for t_i has received bid messages from every $[t_i, a_j] \in \mathcal{B}_i$, that process computes the winning bidder and sends a $next$ message.

These messages are relayed by each process across the graph $\mathcal{G}_p$. While a round refers to a process running $trans$ and $msgs$, a *session* refers to the completion of bidding for each of the N_t tasks. This procedure can be thought of as a distributed Simulated Annealing method.

Let the quantity ND_{ij} be a local estimate of the value of $J(TA)$. This value is updated every time $next$ messages are received by bidding processes where $X_{ij} = 1$, where $i = 1, \ldots, N_t$ and $j = 1, \ldots, N_a$. Let q_{ij} be a random variable with probability density function (pdf),

$$pdf(q_{ij}) = \begin{cases} exp[-\frac{q_{ij}^2}{\sigma_{ij}^2}], q_{ij} > 0, & \text{for } \sigma_{ij} > 0, \\ 0, & \text{for } \sigma = 0 \end{cases}, \quad (30)$$

with standard deviation,

$$\sigma_{ij} = \frac{ND_{ij} \cdot c}{T}, \quad (31)$$

where the constant c is a tuning parameter used to control the rate of collapse of the distribution and time T is a discrete counter equal to the number of sessions. A large value of c increases the probability of finding a solution. However, a small value of c decreases the time needed to find a solution. An in-depth discussion of choosing the value of c is beyond the scope of the current paper. The bid values for each assignable process $[t_i, a_j]$ are computed as,

$$bid_{ij} = nc_{ij} - q_{ij}. \quad (32)$$

where $i = 1, \ldots, N_t$ and $j = 1, \ldots, N_a$.

For the motivational example, consider Table I and round 24 of the bidding procedure. The assignment at round 24 is that of (4), $nc_{11} = 1$, $nc_{14} = 0$, and $ND_{11} = ND_{14}$, resulting in the same distribution for q_{11} and q_{14}. There is a non-zero probability that $[t_1, a_4]$ will outbid $[t_1, a_1]$, but this is unlikely and does not happen. This result is shown in Table I.

The state $state_{ij}$ of process $[t_i, a_j]$ and the messages, $\mathcal{M}$ are defined as follows,

$$\begin{aligned} state_{ij} = (&j, i, \mid \mathcal{C}_i \mid, nc_{ij}, \\ &ND_{ij}, N_{sent_{ij}}, bid_{ij}, X_{ij}, \\ &allBidRecvd_{ij}, allNextRecvd_{ij}, \\ &sendBid_{ij}, sendNext_{ij}), \end{aligned} \quad (33)$$

and

$$\begin{aligned} \mathcal{M}_{bid} = \{(&N_{sent_{ij}}, j, i, \\ &bid_{ij}, \mid \mathcal{C}_i \mid, nc_{ij})\}, \end{aligned} \quad (34)$$

$$\begin{aligned} \mathcal{M}_{next} = \{(&N_{sent_{ij}}, j, i, \\ &X_{ij}, \mid \mathcal{C}_i \mid, nc_{ij})\}, \end{aligned} \quad (35)$$

$$\mathcal{M} = \mathcal{M}_{bid} \cup \mathcal{M}_{next}. \quad (36)$$

where i, j is such that $[t_i, a_j] \in Processes$. The quantities $|\mathcal{C}_i|$, nc_{ij}, ND_{ij} and bid_{ij} are computed per their definitions. The quantity $N_{sent_{ij}}$ is the number of messages sent by process $[t_i, a_j]$. The boolean quantities X_{ij}, $allBidRecvd_{ij}$, $sendNext_{ij}$, and $sendBid_{ij}$ are initialized as zero, and the boolean quantity,

$$allNextRecvd_{ij} = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{otherwise} \end{cases}. \quad (37)$$

Define the function $computeBid$, which computes a bid value from ND_{ij}, nc_{ij}, and T. Also define the function $computeX$, which determines if process $[t_i, a_j]$ is the winning bidder for task t_i. This is done after all bids for task t_i are received by process $[t_i, a_j] \in Processes$. The function $forwardNew$ sends all new incoming messages to all neighbors except for the sending process, and $sendMsg$ sends a message $M \in \mathcal{M}$ to all neighbors. The termination condition for this algorithm is specified as follows:

Termination condition:
$\forall [t_i, a_j] \in Processes$, $ND_{ij} = 0$.

Note that the bid values favor those processes $[t_i, a_j] \in \mathcal{B}_i$ that satisfy more of their connection requirements. There are several properties of the SBA that are of note. Note that, as $J(TA)$ and similarly ND_{ij} decreases, the probability that a process that satisfies a large number of its communication requirements will have winning bids increases. This results in the maximum bid that any process can place for its task occurring when the standard deviation, $\sigma = 0$. This corresponds to $ND_{ij} = 0$, which implies that every process that has won the bidding for its task can communicate with all processes that have won the bidding for the tasks in

```
Data: $M, State_{ij}$
if $allNextRecvd$ then
    $allNextRecvd = 0$
    $computeBid(ND_{ij}, nc_{ij})$
    $sendBid = 1$
end
if $allBidRecvd$ then
    $allBidRecvd = 0$
    $computeX()$
    $sendNext = 1$
end
Result: $state_{ij} \in states_{ij}$
```

Algorithm 1: $trans_{ij}$

```
Data: $state_{ij}$
$forwardNew()$
if $sendBid$ then
    $sendBid = 0$
    $M = M_{bid}$
    $sendMsg(M)$
end
if $sendNext$ then
    $sendNext = 0$
    $M = M_{next}$
    $sendMsg(M)$
end
Result: $M \in \mathcal{M}$
```

Algorithm 2: $msgs_{ij}$

$\mathcal{C}_i$. This implies that if process $[t_i, a_l]$ can match process $[t_i, a_j]$'s bid for task t_i, then process $[t_i, a_l]$ can also meet the same communication requirements. Lemma 6.2 states the correctness of the SBA. The proof of Lemma 6.2 can be found in the Appendix.

Lemma 6.2: Correctness:
If stochastic bidding terminates a feasible assignment, TA, has been found.

VII. Results

The following demonstrates the SBA applied to the motivating example. The example terminates with a feasible assignment in thirty-eight rounds. This corresponds to two complete bidding sessions. Table 1 shows the progression of the bidding process including the values of the various bids. The final assignment for this example is,

$$TA_4 = \{(t_1, a_1), (t_2, a_2), (t_4, a_3), (t_3, a_4)\}. \quad (38)$$

The processes begin by sharing messages for the purpose of initialization, this lasts for seven rounds. Bidding begins at the eighth round. Note the spacing of the rounds between the bidding for each task, this spacing is related to $diam(\mathcal{G}_p)$. The number of rounds required for the sharing of all bid messages is upper bounded by $diam(\mathcal{G}_p)$. The number of rounds required for all the processes bidding on the next task to receive all $next$ messages is also upper bounded by $diam(\mathcal{G}_p)$. This results in a upper bound between the beginning of bidding for t_i and t_{i+1} of $2 \cdot diam(\mathcal{G}_p)$.

Processes only bid for their respective tasks in-turn, but forward messages from other processes at each round. Only the rounds where bid messages are sent are shown in Table I.

TABLE I
Example bidding progress.

Round	Session	$[i,j]$ / bid_{ij}	$[i,l]$ / bid_{il}	Winner
8 (t_1)	1	$[1,1]$ / -4.43	$[1,4]$ / -2.8	a_4
13 (t_2)		$[2,2]$ / -0.14	$[2,3]$ / -1.22	a_2
16 (t_3)		$[3,1]$ / -4.42	$[3,4]$ / -0.28	a_4
21 (t_4)		$[4,2]$ / -2.23	$[4,3]$ / 0.58	a_3
24 (t_1)	2	$[1,1]$ / 0.62	$[1,4]$ / -0.19	a_1
29 (t_2)		$[2,2]$ / 0.92	$[2,3]$ / -0.33	a_2
32 (t_3)		$[3,1]$ / -0.04	$[3,4]$ / 1	a_4
37 (t_4)		$[4,2]$ / 0	$[4,3]$ / 1	a_3

Note that after round 21, the assignment TA is not feasible. This can be seen in Figure 6.

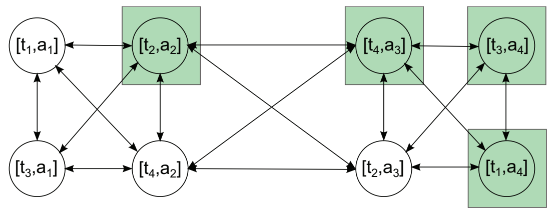

Fig. 6. Session 1 assignments.

The CDAP has been solved after the very next bid (Figure 7), at round 24, processes announce the satisfaction of their local constraints when $next$ messages are sent during the second bidding session. Figure 7 shows the final bid winners.

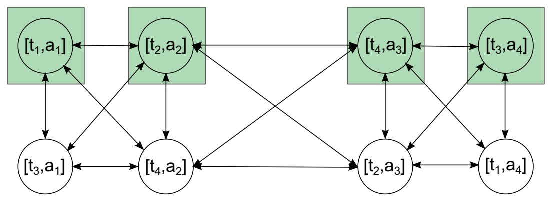

Fig. 7. Session 2 assignments.

VIII. Conclusions and Future Work

This work introduced the Communication-Constrained Distributed Assignment Problem that is important to distributed constrained scheduling applications. The problem was solved using a stochastic bidding method. The algorithm operates in a distributed environment and under the assumption of synchronous communication. The algorithm is correct in the sense that it terminates if and only if a feasible solution to the CDAP is found. The authors are currently extending this work to develop distributed planning algorithms that can operate in the presence of dynamic and faulty communication networks.

IX. Acknowledgements

This research was supported in part by the United States Air Force grant FA 8650-07-2-3744.

References

[1] D. T. Peng, K. G. Shin, and T. F. Abdelzaher, "Assignment and scheduling communicating periodic tasks in distributed real-time systems," *IEEE Transactions on Software Engineering*, vol. 23, pp. 745–757, December 1997.

[2] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous distributed computing systems," *IEEE Concurrency*, pp. 42–51, July 1998.

[3] K. Krishna, K. Ganeshan, and D. J. Ram, "Distributed simulated annealing algorithms for job shop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, pp. 1102–1109, July 1995.

[4] H. W. Kuhn, "The hungarian method for the assignment problem," Bryn Mawr College, Bryn Mawr, PA, Technical Paper, 1955.

[5] S. Baase, *Computer Algorithms: Introduction to Design and Analysis*. Addison-Wesley Publishing Company, 1997, pp. 215–310.

[6] M.Held and R.Karp, "The traveling-salesman problem and minimum spanning trees," IBM Systems Research Institute, Technical Report, 1969.

[7] E. Aarts and J. Lenstra, *Local Search in Combinatorial Optimization*. John Wiley and Sons, 1997, pp. 215–310.

[8] R. Vaessens, E. Aarts, and J. Lenstra, "Job shop scheduling by local search," Eindhoven University of Technology, Eindhoven, Amsterdam, Technical Paper, 1955.

[9] M. Faied, I. Assanein, and A. Girard, "UAV dynamic mission management in adversarial environments," *International Journal of Aerospace Engineering*, pp. 1–10, 2009.

[10] D. Kingston, S. Rasmussen, and M. Mears, "Base defense using a task assignment framework," in *AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, August 2009.

[11] S. Kirkpatrick, J. C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, May 1983.

[12] CAEN HPC Group, "Scheduling jobs on clusters," http://cac.engin.umich.edu/, March 2007.

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

[14] N. Lynch, *Distributed Algorithms*. San Fracisco, California: Morgan Kaufman Publishers, Inc., 1996.

[15] L. Lamport, "Paxos made simple," Citeseer, November 2001.

[16] M. Yokoo, E. H. Durfee, I. Ishida, and K. Kuwabara, "Distributed constraint satisfaction for formalizing distributed problem solving," in *IEEE International Conference on Distributed Systems*, 1992, pp. 614–621.

[17] D. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," Massachusetts Institute of Technology, Cambridge, Massachusetts, Technical Report LIDS, March 1987.

[18] T. Shima, S. Rasmussen, C. Schumacher, N. Ceccarelli, P. Chandler, D. Jacques, B. Kish, and M. Pachter, *UAV Cooperative Decision and Control Challenges and Practical Approaches*. Society for Industrial and Applied Mathematics, 2009.

[19] D. Casbeer, "Decentralized estimation using information consensus filters with a multi-static UAV radar tracking system," Brigham Young University, Provo, Utah, PhD Dissertation, April 2009.

[20] W. Ren, "Multi-vehicle consensus with a time-varying reference state," *Systems and Control Letters*, vol. 56, pp. 474?–483, March 2007.

[21] S. Karaman and G. Inalhan, "Large-scale task/target assignment for uav fleets using a distributed branch and price optimization scheme," in *Proceedings of the 17th World Congress The International Federation of Automatic Control*, Seoul, Korea, July 2008, pp. 13 310–13 317.

[22] P. Dewan and S. Joshi, "Auction-based distributed scheduling in a dynamic job environment," *International Journal of Production Research*, vol. 40, pp. 1173–1191, November 2010.

[23] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Mineola, New York: Dover, 1982.

X. Appendix

Proof: Proof of Proposition 4.1
When choosing which agent to assign to a task, there are N_a choices. There are N_t tasks. Hence there are $\mathcal{O}(N_a^{N_t})$ choices to for the assignments in the worst case. ■

Proof: Proof of Proposition 4.2:
Consider that the possible task assignments $(t_i, TA(t_i))$ can be represented by a matrix $X \in \{0,1\}^{N_t \times N_a}$, where $X_{ij} = 1$ if the task t_i is assigned to agent a_j and $X_{ij} = 0$ otherwise. Let the matrix B_c represent the adjacency matrix of the graphs $\mathcal{G}_c$. That is, $B_{c_{jl}} = 1$ if $\{a_i, a_l\} \in \mathcal{E}_c$, $B_{c_{jl}} = 0$ otherwise. Let the matrix B_{cl} represent the clustering relationships between tasks. That is, $B_{cl_{ik}} = 1$ if there exists a cluster $\mathcal{T}_m$ such that $t_i \in \mathcal{T}_m$ and $t_k \in \mathcal{T}_m$.

The matrix product $X^T B_p X \in \{0,1\}^{N_a \times N_a}$ has the following meaning. The matrix $(X^T B_{cl} X)_{jl} = 1$ if the agents a_j and a_l are assigned tasks that share a cluster. The matrix $(X^T B_{cl} X)_{jl} = 0$ otherwise.

Let the matrix AC represent the capability feasibility constraints of $Capability$. Here, $AC_{ij} = 0$ if $(t_i, a_j) \in Capability$ and 1 otherwise. The following constraints must be satisfied in order for the task assignment of X to be feasible.

$$B_{N_{il}} - (XB_pX^T)_{il} \geq 0, \quad \forall i, l = 1, \ldots, N_a, \tag{39}$$

$$\sum_{j=1}^{N_v} AC_{ij} X_{ij} = 0, \quad \forall i = 1, \ldots, N_a, \tag{40}$$

$$\sum_{i=1}^{N_a} X_{ij} = 1, \quad \forall j = 1, \ldots, N_v. \tag{41}$$

Equation 39 is nonlinear in the assignments. This proves Proposition 4.2.

■

Proof: Proof of Proposition 6.1
"$\rightarrow$" Assume (15) is satisfied. This implies that for all $\{t_i, t_k\} \in \mathcal{T}_m, m = 1, \ldots, N_{cl}$, $\{TA(t_i), TA(t_k)\} \in \mathcal{E}_c$. Hence, for all pairs $(t_i, TA(t_i)), i = 1, \ldots, N_t$, and for all processes $[t_i, a_j]$ where $X_{ij} = 1$, $nc_{ij} = \mid \mathcal{C}_i \mid$. This implies that $J(TA) = 0$.

"$\leftarrow$" Assume $J(TA) = 0$, this implies that for all processes $[t_i, a_j]$ where $X_{ij} = 1$, $nc_{ij} = \mid \mathcal{C}_i \mid$. Hence, for all $\{t_i, t_k\} \in \mathcal{T}_m, m = 1, \ldots, N_{cl}$, $\{TA(t_i), TA(t_k)\} \in \mathcal{E}_c$. That is, (15) is satisfied.

■

Proof: Proof of Lemma 6.2
Assume bidding has terminated, that is $ND_{ij} = 0$ for all processes. $ND_{ij} = 0$ for all processes implies that for all processes, $nd_{ij} \cdot X_{ij} = 0$. By the definition of nd_{ij} and X_{ij}, and Proposition 6.1, TA is feasible.

■