Decentralized Planning for Complex Missions with Dynamic Communication Constraints

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

Decentralized Planning for Complex Missions with Dynamic Communication Constraints

Sameera Ponda, Josh Redding, Han-Lim Choi, Jonathan P. How, Matt Vavrina and John Vian

Abstract— **This paper extends the consensus-based bundle algorithm (CBBA), a distributed task allocation framework previously developed by the authors, to address complex missions for a team of heterogeneous agents in a dynamic environment. The extended algorithm proposes appropriate handling of time windows of validity for tasks, fuel costs of the vehicles, and heterogeneity in the agent capabilities, while preserving the robust convergence properties of the original algorithm. An architecture to facilitate real-time task replanning in a dynamic environment is also presented, along with methods to handle varying communication constraints and dynamic network topologies. Simulation results and experimental flight tests in an indoor test environment verify the proposed task planning methodology for complex missions.**

I. Introduction

Modern day network centric operations involve large teams of agents, with heterogeneous capabilities, interacting together to perform missions. These missions involve executing several different tasks such as conducting reconnaissance, surveillance, target classification, and rescue operations [1]. Within the heterogeneous team, some specialized agents are better suited to handle certain types of tasks than others. For example, UAVs equipped with video can be used to perform search, surveillance and reconnaissance operations, human operators can be used for classification tasks, ground teams can be deployed to perform rescue operations, etc. Ensuring proper coordination and collaboration between agents in the team is crucial to efficient and successful mission execution. To this effect it is of interest to develop autonomous task allocation methods to improve mission coordination.

Autonomous task allocation is a significantly complex combinatorial problem (NP-hard) [6]. The planner must simultaneously allocate a set of tasks among a team of heterogeneous agents so as to optimize overall mission efficiency and reduce expected costs. These tasks can have different locations and time-windows of validity and may require coordinated execution between several agents [4,10]. Furthermore, the planning architecture must account for vehicle limitations, agent-task compatibility requirements, and network configuration and communication requirements. Fig. 1 depicts the structure of a real-time autonomous task allocation architecture for a heterogeneous team. The overall system involves a mission control center responsible for defining a list of tasks that comprise the mission, a decentralized network detection scheme to detect the network/sub-network structure in real-time, decentralized task allocation planners that coordinate planning within each sub-network, assigning tasks to the different agents, and sensors and actuators to interact with the "World". Given up-to-date agent models, network configuration information, and task lists, the decentralized planning process allocates the resources (i.e. agents) to the respective tasks over some planning horizon, thereby creating schedules for each of the heterogeneous agents. This allocation is determined by taking into account the availability and capabilities of the agents up front, with the object of minimizing delays and costs while improving mission efficiency. The planning loop is executed in real-time and the agent models and network configuration are updated as more information from the world becomes available.

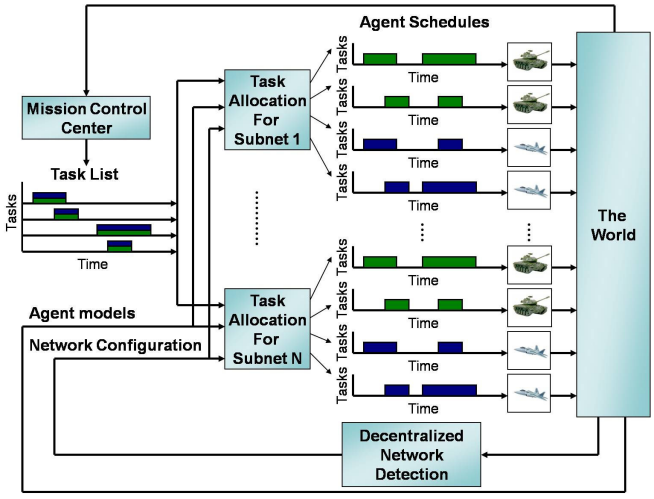

Fig. 1. Real-time decentralized task allocation architecture for a heterogeneous team divided into sub-networks

Previous literature on multi-agent multi-task allocation has focused on many variants of the Traveling Salesman Problem (TSP) and the Dynamic Vehicle Routing Problem (DVRP). Extensive literature has focused on problems involving DVRP with time-windows (DVRPTW) [18]. A few examples include servicing impatient customers [16], where a centralized planning framework is used to distribute resources to customers with strict service time-windows, and developing MILP frameworks and hybrid models for DVRPTW problems [10,14]. Other work has focused on multi-agent coordination in dynamic environments with applications for teams of UAVs [4,7,8]. Many of these approaches involve solving large, complex combinatorial optimization problems to allocate resources and coordinate the behavior of multiple

S. Ponda, J. Redding, H.-L. Choi, and J. P. How are with the Dept. of Aeronautics and Astronautics, MIT, Cambridge, MA, {sponda, jredding, hanlimc, jhow}@mit.edu

M. Vavrina and J. Vian are with Boeing R&T, Seattle, WA, {matthew.a.vavrina, john.vian}@boeing.com

heterogeneous agents.

For large teams of agents, centralized approaches quickly become infeasible and decentralized architectures must be adopted [8]. One class of decentralized combinatorial algorithms involves using auction algorithms augmented with consensus protocols to allocate tasks over a team of agents while resolving conflicting assignments locally among the agents [2,7,8,17]. Most of this previous multi-agent multi-task allocation work does not consider team heterogeneity, dynamic environments or varying communication constraints. Literature on limited communication environments includes several examples of maintaining network connectivity through relay teams [12,15] and optimally placing vehicles in order to maintain communication links [9]. While these works address the challenges of dealing with communications in uncertain and dynamic environments, they restrict networks of agents to always maintain connectivity, limiting the scope of the theater of operation. In realistic mission scenarios, communication links are broken and created in real-time and teams of agents can reconfigure themselves into different network/sub-network structures.

This paper presents a decentralized task allocation algorithm for a network of heterogeneous agents that simultaneously allocates tasks with known time-windows of validity to all agents in the heterogeneous team. A real-time replanning architecture is also implemented to handle changes in the environment and varying communication constraints, and different strategies to execute distributed planning over sub-networks are proposed and compared.

II. Distributed Tasking with Time Windows

A. Problem Statement

Given a list of N_t tasks and N_a agents, the goal of the task allocation algorithm is to find a conflict-free matching of tasks to agents that maximizes some global reward. An assignment is said to be free of conflicts if each task is assigned to no more than one agent. The global objective function is assumed to be a sum of local reward values, while each local reward is determined as a function of the tasks assigned to that particular agent.

The task assignment problem described above can be written as the following integer (possibly nonlinear) program:

$$\begin{aligned} \max \quad & \sum_{i=1}^{N_a} \left(\sum_{j=1}^{N_t} c_{ij}(\tau_{ij}(\mathbf{p}_i(\mathbf{x}_i)))x_{ij} \right) \\ \text{subject to:} \quad & \sum_{j=1}^{N_t} x_{ij} \leq L_t, \ \forall i \in \mathcal{I} \\ & \sum_{i=1}^{N_a} x_{ij} \leq 1, \ \forall j \in \mathcal{J} \\ & x_{ij} \in \{0,1\}, \ \forall (i,j) \in \mathcal{I} \times \mathcal{J} \end{aligned} \tag{1}$$

where the binary decision variable x_{ij} is 1 if agent i is assigned to task j, and $\mathbf{x}_i \in \{0,1\}^{N_t}$ is a vector whose j-th element is x_{ij}. The index sets are defined as $\mathcal{I} \triangleq \{1,\ldots,N_a\}$ and $\mathcal{J} \triangleq \{1,\ldots,N_t\}$. The vector $\mathbf{p}_i \in (\mathcal{J} \cup \{\emptyset\})^{L_t}$ represents an ordered sequence of tasks for agent i; its k-th element is $j \in \mathcal{J}$ if agent i conducts j at the k-th point along the path, and becomes $\emptyset$ (denoting an empty task) at the k-th point if agent i conducts less than k tasks. L_t is a limit on the maximum number of tasks that can be assigned to an agent. The summation term in brackets in the objective function represents the local reward for agent i.

Key assumptions underlying the above problem formulation are:

1) The score c_{ij} that agent i obtains by performing task j is defined as a function of the arrival time τ_{ij} at which the agent reaches the task (or possibly the expected arrival time in a probabilistic setting).
2) The arrival time τ_{ij} is *uniquely* defined as a function of the path $\mathbf{p}_i$ that agent i takes.
3) The path $\mathbf{p}_i$ is *uniquely* defined by the assignment vector of agent i, $\mathbf{x}_i$.

Many interesting design objectives for multi-agent decision making problems feature scoring functions that satisfy the above set of assumptions. The time-discounted value of targets [3,5] is one such example, in which the sooner an agent arrives at the target, the higher the reward it obtains. However, in more complex missions scenarios, it may not be desirable to visit the target as soon as possible. For example, if the task is to re-investigate a previously observed target at some scheduled time in the future, a more reasonable choice of score function would have its maximum at the desired re-visiting time and lower values at re-visit times around the optimal time. This work develops methodologies to address these types of complicated scoring structures.

B. Consensus-Based Bundle Algorithm

Our approach to this complex combinatorial optimization planning problem is inspired by the Consensus-Based Bundle Algorithm (CBBA) [8]. CBBA is a distributed auction protocol that provides provably good approximate solutions for multi-agent multi-task allocation problems over networks of agents. 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. There are several core features of CBBA that can be exploited to develop an efficient planning mechanism for heterogeneous teams. First, CBBA is a decentralized decision architecture, which is a necessity for planning over large teams due to the increasing communication and computation overhead required for centralized planning with a large number of agents. Second, CBBA is a polynomial-time algorithm. The worst-case complexity of the bundle construction is $\mathcal{O}(N_t L_t)$ and CBBA converges within $\max\{N_t, L_t N_a\}D$ iterations, where D is the network diameter (always less than N_a). Thus, the CBBA framework scales well with the size of the network and/or the number of tasks (or equivalently, the length of the planning horizon). Third, various design objectives, agent models, and constraints can be incorporated by defining appropriate scoring functions. If the resulting

scoring scheme satisfies a certain property called *diminishing marginal gain* (DMG) [8], a provably good feasible solution is guaranteed. The next section describes the extensions to this algorithm to explicitly account for tasks with time-windows of validity, and addresses a few implementation details for using CBBA to plan for heterogeneous teams of autonomous vehicles.

C. Scoring Functions with Time Windows

Definition 1 To begin the discussion on incorporating scoring functions with more complicated temporal dependencies, this work defines the following entities:

1) *Score Profile ($s_j(t)$):* The score profile $s_j(t)$ represents the reward an agent gets from task j when it arrives at the task at time t, and is based on the value of the task, R_j, and any time penalty associated with the task. An example score profile is $s_j(t) = e^{-\lambda_j(t-t_{j_{start}})}R_j$, where $(t - t_{j_{start}})$ is the difference between the task start time and the agent arrival time, and $\lambda_j > 0$ is a discount parameter to penalize late arrivals. Without time discounting the score profile is $s_j(t) = R_j$.
2) *Time Window ($u_j(t)$):* The time window of validity for a task represents the time in which the task is allowed to be started. For task j this window is defined as

$$u_j(t) = \begin{cases} 1, & t_{j_{start}} \leq t \leq t_{j_{end}} \\ 0, & \text{otherwise.} \end{cases}$$

Using time windows for tasks provides a framework to penalize early arrivals as well as late arrivals.

The score an agent receives for a task is a function of his arrival time at the task location, τ_{ij}, and can be computed as $c_j(\tau_{ij}) = s_j(\tau_{ij})u_j(\tau_{ij})$. The arrival time, τ_{ij}, is in turn a function of the path the agent has taken before reaching task j. Given a path $\mathbf{p}_i$ which is composed of tasks, and a corresponding set of best times $\tau^\star_{ik}(\mathbf{p}_i)$ for all $k \in \mathbf{p}_i$, the bidding process can be described as follows. For each task $j \notin \mathbf{p}_i$, the best time to do task j can be found by solving the following problem,

$$\begin{aligned} \tau^\star_{ij}(\mathbf{p}_i) &= \underset{\tau_{ij}\in[0,\infty)}{\operatorname{argmax}}\ c_j(\tau_{ij}(\mathbf{p}_i \oplus j)) \\ \text{subject to: } & \tau_{ik}(\mathbf{p}_i \oplus j) = \tau^\star_{ik}(\mathbf{p}_i), \quad \forall k \in \mathbf{p}_i \end{aligned} \tag{2}$$

where $\oplus$ signifies inserting task j into path $\mathbf{p}_i$ without shuffling the order of tasks already in $\mathbf{p}_i$. The constraint states that the insertion of the new task j into path $\mathbf{p}_i$ cannot impact the current arrival times for the tasks already in the path. The path is updated by inserting j in the best location, $\mathbf{p}_i \leftarrow (\mathbf{p}_i \oplus j)$. The best time and score for task j are then saved as $\tau_{ij}(\mathbf{p}_i) = \tau^\star_{ij}$ and $c_{ij}(\tau_{ij}(\mathbf{p}_i)) = c_j(\tau^\star_{ij})$.

An important property for convergence is the *diminishing marginal gain* property (DMG). In words, DMG means that the score for a task not in the path cannot increase as more tasks are added to the path, i.e., $\forall j \notin \mathbf{p}_i$

$$c_{ij}(\tau_{ij}(\mathbf{p}_i \oplus j)) \geq c_{ij}(\tau_{ij}(\mathbf{p}'_i \oplus j))$$

where $\mathbf{p}'_i = \{\mathbf{p}_i \oplus m\}$.

Consider the calculation of the best arrival time for task j when the current path is $\mathbf{p}'_i$ instead of $\mathbf{p}_i$. Then, the following optimization needs to be solved:

$$\begin{aligned} \tau^\star_{ij}(\mathbf{p}'_i) &= \underset{\tau_{ij}\in[0,\infty)}{\operatorname{argmax}}\ c_j(\tau_{ij}(\mathbf{p}'_i \oplus j)) \\ \text{subject to: } & \tau_{ik}(\mathbf{p}'_i \oplus j) = \tau^\star_{ik}(\mathbf{p}'_i), \quad \forall k \in \mathbf{p}'_i \end{aligned} \tag{3}$$

The constraint can be rewritten recursively as the following set of constraints,

$$\tau_{ik}(\mathbf{p}_i \oplus m \oplus j) = \tau^\star_{ik}(\mathbf{p}_i \oplus m) = \tau^\star_{ik}(\mathbf{p}_i), \quad \forall k \in \mathbf{p}_i \tag{4}$$

$$\tau_{im}(\mathbf{p}_i \oplus m \oplus j) = \tau^\star_{im}(\mathbf{p}_i \oplus m). \tag{5}$$

Therefore, calculation of $\tau^\star_{ij}(\mathbf{p}'_i)$ involves solving an optimization with the same objective function but an additional constraint (5). Thus, the optimal objective value for (3) cannot be greater than that for (2); i.e. $c_j(\tau^\star_{ij}(\mathbf{p}_i)) \geq c_j(\tau^\star_{ij}(\mathbf{p}'_i))$, which means the DMG property is satisfied. In other words, with the arrival time defined by the optimization in (2), the score function satisfies DMG regardless of the details of the score profiles.

D. CBBA Implementation Details

A few other implementation details for defining the problem in CBBA involve accounting for agent-task compatibility and for expected fuel consumption while executing the task. In order to account for the heterogeneous nature of the team, agents can be classified according to their capabilities and tasks according to their requirements. A set of constraints can then be incorporated into the planning process specifying which types of agents can do which types of tasks (i.e. UAV's can perform aerial surveillance, ground teams can perform rescue operations, etc). A straightforward way to incorporate these constraints is by making the agent's bid zero for tasks with which it is not compatible.

To account for fuel consumption, the score function for a task can be augmented with a fuel penalty due to travel distance for a given agent,

$$c_{ij}(\tau^\star_{ij}, \mathbf{p}_i) = e^{-\lambda_j(\tau^\star_{ij} - t_{j_{start}})}R_j u_j(\tau^\star_{ij}) - F_i \Delta D_{ij}(\mathbf{p}_i)$$

where F_i is the cost of fuel per meter incurred by agent i and $\Delta D_{ij}(\mathbf{p}_i)$ is the distance traveled by the agent to get to the task location from its previous location. To ensure satisfaction of DMG, a constant heuristic distance, $\Delta D'_{ij}$, representing the distance from the vehicle's *initial position* to the task location, was used instead. Monte Carlo simulation results verified that this type of heuristic penalty produced equally efficient task allocation assignments as those obtained using the actual travel distance $\Delta D_{ij}(\mathbf{p}_i)$, while guaranteeing convergence of the algorithm to conflict-free assignments.

III. Distributed Planning in an Uncertain and Dynamic Environment

A. Real-Time Replanning Architecture

In order to ensure that the task allocation remains relevant in a dynamically changing environment it is necessary to

replan in real-time. Replanning at a fast enough rate ensures that vehicle states and network topologies are up to date, new tasks are accounted for and older or irrelevant tasks are pruned, and that the impact of discrepancies between the agent models in the planner and the actual agent behavior is minimized. Figure 1, shown in Section I, depicts the overall system architecture. The vehicle models are updated in real-time using vehicle states from the world, decentralized network detection is performed locally by agents through communication with their immediate neighbors, and the task list is maintained by adding new pop-up tasks and pruning completed tasks. The "World" can represent either a simulated world or a real flight test environment as described in Section III-D. The decentralized planning algorithm runs in real-time and leads to a deconflicted task allocation if the network maintains connectivity. If the network becomes disconnected, agents will not know about other agents' bids outside of their sub-network, nor will they be able to communicate with these other agents in order to execute consensus. In this situation the planner may not converge and multiple agents from different sub-networks might bid on the same tasks leading to conflicting assignments. The next section describes these communication challenges and proposes methodologies for handling network disconnects.

B. Dynamic Network Handling Protocols

In a dynamic mission environment, it is likely that some communication links may break and other new ones may be formed. For example, if vehicles need to be within a certain distance in order to communicate (communication radius), but if there exist some tasks such that a vehicle is forced to travel outside of this communication radius, then the vehicle must lose connectivity with its neighbors in order to accomplish these tasks. In these situations, since the vehicle is not able to communicate its current winning bids, the next round of replanning may assign that agent's tasks to other agents. This is undesirable since sending multiple agents to do the same tasks leads to unnecessary fuel consumption. Furthermore, it is assumed that when vehicles get within the prescribed communication radius they will be able to resolve the assignment conflict, but if the planner replan rate is not fast enough, they may not be able to deconflict in time, possibly leading to collisions. It is necessary, therefore, to have a method to ensure that task assignments remain conflict-free in the presence of network disconnects.

There are several possible methods of handling varying network topologies. Depending on the bandwidth, level of detail, and availability of communications with the mission control center, different methods for distributing tasks amongst the various sub-networks can be adopted. This work compares a few of these, discussing the advantages and drawbacks of each. First we consider the default behavior, where all agents know about and are free to bid on any tasks in the entire task list (No task list adjustment). If the network is disconnected, the task allocation algorithm will run locally within each sub-network, and the global allocation may contain conflicting assignments between agents in different sub-networks. Next, we consider a deconfliction protocol that requires that the mission control center distribute all the tasks amongst the agents by assigning tasks to the closest compatible agent to the particular task (Central task list adjustment). Agents then run the task allocation algorithm locally within their sub-network over the set of tasks that are assigned to agents in that sub-network. This approach guarantees conflict-free assignments but requires the mission control center to redistribute the entire task list amongst the agents every time a replan is required, which for realistic missions would involve significant overhead and communication, limiting the real-time performance of the team. A last approach considered involves notifying only the closest compatible agent to the task every time a new task is created, and replanning locally within each sub-network over the set of tasks that are currently in the paths and in the new task lists for all agents within that sub-network (Local task list adjustment), which also guarantees conflict-free assignments. Both the central and local task list adjustment methods require that the mission control center maintain updated agent information, however, the local adjustment method involves significantly less overhead than the central method, since task assignments from the mission control center only occur once per new task. The next section describes simulation results comparing these three different methods.

C. Comparison of Different Network Handling Protocols

The scenario used to test the different task adjustment approaches described above involved a team of 12 heterogeneous agents (6 UAVs and 6 ground robots). The simulation was initialized with 12 UAV tasks with random start times, 40 sec time windows and 5 sec durations. Once the UAV tasks were started a secondary ground robot rescue task was created for each UAV task. Additional pop-up UAV tasks were created at 5 sec intervals and the task allocation algorithm replanned every 2 sec. The simulation consisted of a mission control center, a network detector and local agent simulations. The network detector used the vehicle positions and a communication radius parameter to determine if two vehicles were able to communicate and returned a list of subnetworks. The local agent simulations implemented models of the vehicles to execute the tasks in each agent's path. The mission control center maintained the list of tasks by creating pop-up tasks and pruning completed tasks from the list, in addition to implementing the responsibilities for the different task adjustment methods described in the previous section. The overall mission score was obtained by adding the individual scores for the agents as described in Section II-D

Figure 2 presents a snapshot of the simulation interface showing the agent paths and schedules over the course of the simulation. The display on the left shows the agents and their proposed paths and the tasks along with a countdown to their expiry time. The right display shows the agent schedules, including the past history and the proposed future schedule (on either side of the time line). The time-windows of validity for the tasks are shown (black lines) along with the actual

time that the agent executed the task (colored block).

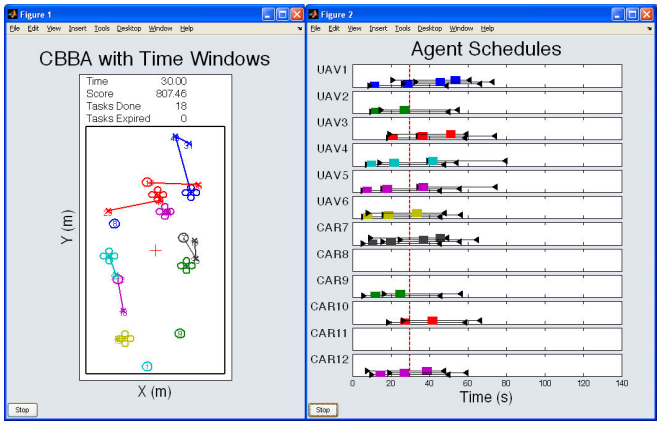

Fig. 2. Simulation showing 12 agents (6 UAVs & 6 UGVs) bidding on and accomplishing a dynamic set of tasks

Using this simulation infrastructure as a testbed, the three methods described in the previous section were implemented, and Monte Carlo simulations of 200 iterations were executed to compare the mission performance for these three approaches under different communication radii. Figure 3 shows the overall mission scores, the number of completed tasks and the team fuel consumption as a function of the communication radius normalized by the maximum distance of the theater of operation. The results show that for all three methods the mission score increases as the communication radius increases, since agent coordination improves with communication. With a normalized communication radius of about 0.3 and higher and with a team of 12 agents, the network remains connected in most cases and all three methods yield similar performance. With less agents this communication radius threshold would be higher since, for a given communication radius, it is more likely that the network would lose connectivity with less agents. The baseline case (No adjustment) is seen to have the lowest score and highest fuel consumption, especially at low communication radii. This is because without task list adjustments there will be many assignment conflicts between different subnetworks, resulting in unnecessary fuel usage from having multiple agents attempt to perform the same tasks as well as a lower number of overall completed tasks (since agents are busy traveling to tasks that they will never accomplish). As the connectivity decreases and the number of subnetworks increases this problem becomes worse. With task list adjustments the mission performance greatly improves as seen in the results for both the central and local task list adjustment methods. Since the task allocation is guaranteed to be conflict-free over the entire team there is no excess fuel usage and the total number of completed tasks is higher since the coordination of the team is improved. The central adjustment method has lower total fuel consumption than the local adjustment method, however, due to the amount of overhead required by the mission control center this strategy does not scale well as the size of the team increases. The local adjustment method achieves a similar number of

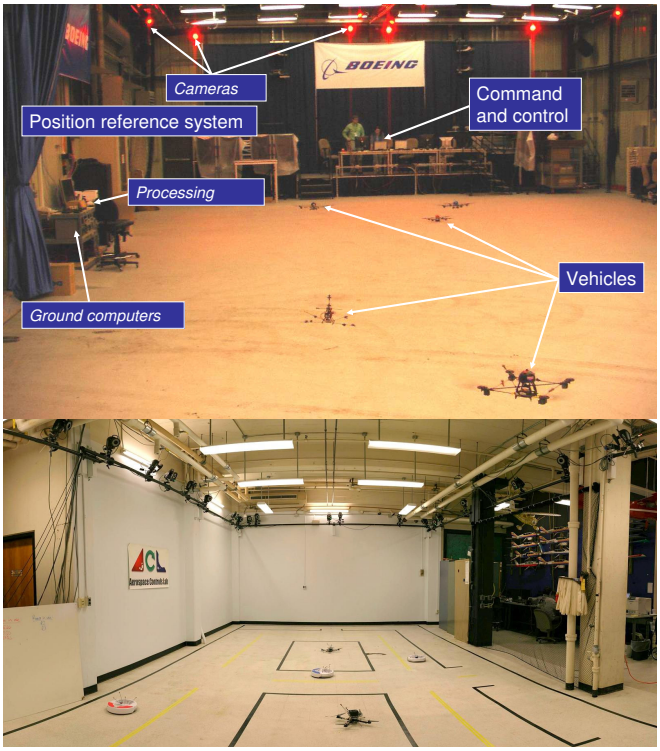

Fig. 4. Boeing Vehicle Swarm Technology Laboratory [11] (top) and MIT RAVEN Multi-Vehicle Testbed [19] (bottom)

completed tasks as the central adjustment method, and although the fuel usage is slightly higher, the overhead required to implement this local adjustment strategy is significantly lower. The next section describes the implementation of real-time decentralized planning within local sub-networks on actual vehicles in an indoor flight facility.

D. Complex Mission Execution in a Real Flight Environment

Flight experiments were conducted at the Boeing Vehicle Swarm Technology Laboratory (VSTL) [11] and at MIT's *Real-time indoor Autonomous Vehicle test ENvironment* (RAVEN) [13,19], shown in Figure 4. These indoor flight facilities are equipped with motion-capture systems which yield accurate, high-bandwidth position and attitude data for all tracked vehicles within the flight volume. The environmental conditions of these indoor facilities can be carefully controlled for flight testing, and can range from ideal to wind-induced. The controlled environment is available 24/7 since it is not dictated by weather, poor visibility, day/night cycles, or other external factors.

Flight experiments were conducted for a heterogeneous team of 6 agents (3 quad-rotor air vehicles and 3 ground vehicles), with a normalized communication radius of 0.1. CBBA with time-windows was used to perform the task allocation within the sub-networks and the different replanning architectures with task list adjustments described in the previous sections were implemented. The flight results, shown in Table I, exhibit similar trends to those shown in the simulation results. Both the central and local adjustment methods achieved similar scores and number of

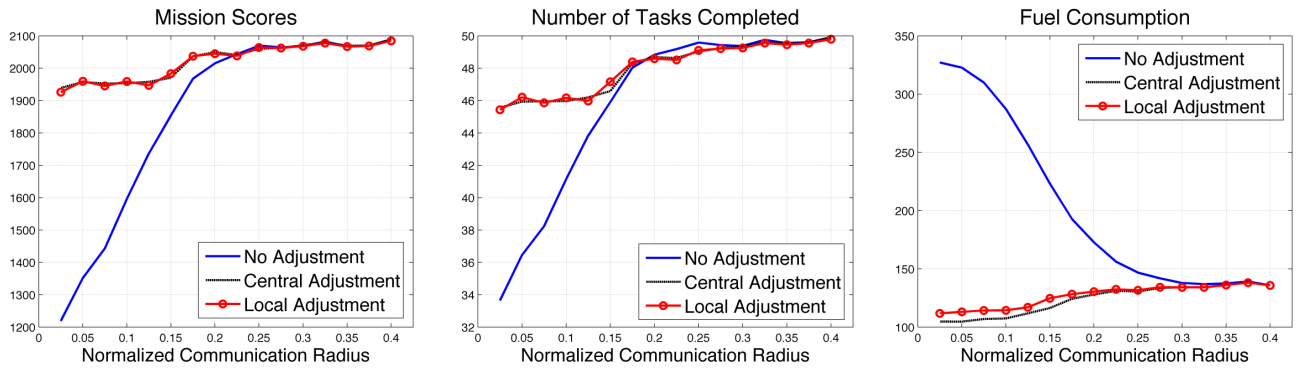

Fig. 3. Mission scores, completed tasks and fuel consumption as a function of communication radius for different network handling protocols

TABLE I
FLIGHT TEST RESULTS

Adjustment Method	Score	Tasks	Fuel
No Adjustment	897.32	22	111.35
Central Adjustment	1561.44	37	62.79
Local Adjustment	1458.46	34	71.51

tasks completed. The central adjustment method performed slightly better than the local adjustment method, with a lower overall fuel consumption as expected, but with a higher computational and communication overhead. With no task list adjustments the team performance was fairly poor with more fuel consumed and less overall tasks completed.

IV. CONCLUSION

This paper presents a real-time decentralized task allocation algorithm for a team of heterogeneous agents operating in a complex environment. The algorithm described is an extension of the CBBA planning algorithm, a polynomial-time decentralized auction protocol that provides provably good approximate solutions for multi-agent multi-task allocation problems. This paper extends the CBBA algorithm to explicitly account for tasks with time-windows of validity and heterogeneous agent-task compatibility constraints. A real-time replanning architecture is presented along with task list adjustment methods to handle dynamic network topologies, and advantages and drawbacks for the different methods are discussed. Implementing local task list adjustments is shown to drastically improve mission performance under low communication environments, with only marginal increases in required overhead. Simulation and experimental flight tests have verified that this decentralized algorithm can successfully enable a heterogeneous team to perform complex missions in real-time dynamic environments under varying communication constraints.

ACKNOWLEDGMENTS

This research was supported in part by AFOSR (FA9550-08-1-0086), MURI (FA9550-08-1-0356), and Boeing Research and Technology.

REFERENCES

[1] Unmanned aircraft systems roadmap, 2005-2030. Technical report, Office of the Secretary of Defense, August 2005.

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

[3] M. Alighanbari and J. P. How. Decentralized task assignment for unmanned aerial vehicles. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, 2005.

[4] M. Alighanbari, Y. Kuwata, and J. How. Coordination and control of multiple uavs with timing constraints and loitering. In *American Control Conference, 2003. Proceedings of the 2003*, volume 6, pages 5311–5316 vol.6, June 2003.

[5] 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.

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

[7] L. Bertuccelli, H. Choi, P. Cho, and J. How. Real-time Multi-UAV Task Assignment in Dynamic and Uncertain Environments.

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

[9] C. Dixon and E. Frew. Maintaining optimal communication chains in robotic sensor networks using mobility control. *Mobile Networks and Applications*, 14(3):281–291, 2009.

[10] R. Dondo and J. Cerdá. An MILP framework for dynamic vehicle routing problems with time windows. *Latin American Applied Research*, 36(4):255–261, 2006.

[11] E. Saad, J. Vian, G.J. Clark and S. Bieniawski. Vehicle Swarm Rapid Prototyping Testbed. In *AIAA Infotech@Aerospace*, Seattle, WA, 2009.

[12] A. Ibrahim, K. Seddik, and K. Liu. Improving connectivity via relays deployment in wireless sensor networks. In *Proc. IEEE Global Telecommunications Conference (Globecom07)*, pages 1159–1163, 2007.

[13] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *Control Systems Magazine*, 28(2):51–64, April 2008.

[14] Q. Jun, J. Wang, and B. Zheng. A Hybrid Multi-objective Algorithm for Dynamic Vehicle Routing Problems. *Lecture Notes in Computer Science*, 5103:674–681, 2008.

[15] H. Nguyen, N. Pezeshkian, M. Raymond, A. Gupta, and J. Spector. Autonomous communication relays for tactical robots. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, 2003.

[16] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler. A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *Mobile Networks and Applications*, 14(3):350–364, 2009.

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

[18] P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial Mathematics, 1987.

[19] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron. Indoor Multi-Vehicle Flight Testbed for Fault Detection, Isolation, and Recovery. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Keystone, CO, August 2006.