Multiple UAV Task Allocation using Negotiation

P.B. Sujit
Department of Aerospace Engineering
IISc, Bangalore, India
sujit@aero.iisc.ernet.in

A. Sinha
Department of Aerospace Engineering
IISc, Bangalore, India
asinha@aero.iisc.ernet.in

D. Ghose
Department of Aerospace Engineering
IISc, Bangalore, India
dghose@aero.iisc.ernet.in

ABSTRACT

A multiple UAV search and attack mission involves searching for targets in an unknown region, followed by attack on the detected targets. An effective mission involves assigning the tasks to UAVs efficiently. Task allocation becomes difficult when the UAVs have limited range to detect the targets and neighbouring UAVs. The UAVs are also subject to limited communication range. With these constraints, allocating tasks efficiently to UAVs become difficult. In this paper, we propose a negotiation scheme that efficiently allocates tasks for multiple UAVs. The performance of the negotiation based task allocation is studied on a battle field scenario. We study the effect of sensor ranges on the task allocation and compare the results with that of greedy strategy and a variation of the negotiation mechanism. The results show that the negotiation based task allocation mechanism performs far better than the greedy strategy. Negotiation with target information exchange between neighbours performs better than without target information exchange, when the sensor radius is low.

Keywords

Task allocation, Negotiation, Multiple agents, UAV search and attack tasks

1. INTRODUCTION

Unmanned aerial vehicles (UAVs) are being extensively used for military purposes, like search, surveillance and as munitions in the battlefield [1] - [10]. They play a crucial role in information gathering from hostile and unknown regions. These UAVs can also be used as munitions to search, attack and destroy targets in the unknown region. The UAVs used for these applications may have limited capabilities and may not have the required stealth capability and ammunition payload to complete the task single-handedly. Hence, there is a necessity for such UAVs to be deployed in swarms. A desirable feature for these UAV swarms is to have autonomous decision making and coordinating capability. As the UAVs perform search, they may find several targets in the search region. The number of UAVs may be more than the number of targets available or the number of targets available may be more than the number of UAVs. In either case, we need an efficient task allocation method for assigning the UAVs to the targets. The classical solution for task allocation problem is to have a centralized task allocation system that generates the necessary commands for the UAVs. But, centralized task allocation system have well known limitations and do not address scalability issues well. Hence, there is a necessity to develop a decentralized task allocation algorithm. This algorithm must be suitable for implementation in a multiple agent UAV swarm. It should also be scalable, and consume low computational overhead. An efficient task allocation strategy should have the ultimate objective to complete the mission (that is, destroy all targets) in minimum time by cooperating and coordinating with other UAVs [1]. Cooperation can be achieved by communication with neighbouring UAVs, explicitly or implicitly. Here, we use negotiation mechanism to develop such a decentralized task allocation algorithm for multiple UAVs that communicate with each other for decision-making while performing search and attack tasks in an unknown region.

The UAVs that we consider are small in size, have limited fuel capacity (limited flight time), and limited sensing capabilities. The UAVs can detect the presence of neighbouring UAVs and targets within its sensor range only. The UAVs communicate with their neighbours and make their decisions based only on the information they receive from their sensors. All the UAVs are assumed to be homogeneous, have constant speed, and have no turn radius constraints. The UAVs have to carry out the mission within the given flight time. Collision avoidance between UAVs is also not an issue.

The UAVs can perform search, attack, and speculative tasks. Search task refers to searching for targets in the unknown region. Once a target is found by the UAV, it executes a speculative task to ensure that the target is a real target and not a false target. The target that has been verified as a real target is attacked by the UAV. We assume that once a target is attacked, it is destroyed and hence battle damage assessment task on the target is not necessary to be performed. However, it can be incorporated in the formulation fairly easily.

The speculative task is performed in the following way: Once a UAV finds a target, it estimates the value of the target. This value is used for negotiating with the neighbouring UAVs for assignment of the target.

Each UAV performs decision-making independently, based on the estimate of the target status. The decision taken by the UAV also depends on the number of neighbouring UAVs. Here, we assume that the UAVs do not have sufficient memory to remember the path it has travelled.

1.1 Literature

Task allocation of UAVs is an active research area for the past few years. Nygard et al. [1], propose a network flow optimization model for allocating UAVs to targets. The network optimization problem is formulated as a linear programming problem to obtain decisions for allocating the UAVs. The authors assume that global communication between UAVs exists. The network flow model has been further extensively studied by Schumacher [2, 3, 4] for wide area search munitions with variable path lengths, assignment with timing constraints and path planning. Chandler et al. [5, 6], explore various other techniques like iterative network flow, auctions, linear programming, and mixed integer linear programming, for multi-UAV task allocation. The effect of communication delays on the task allocation using the iterative network flow model is dealt with in Mitchell [7]. Curtis [8] presents a task allocation methodology for simultaneous search and target assignment, where the search and task assignments are posed as a single optimization problem. Turra et al. [9] present a task allocation algorithm for multiple UAVs performing search, identification, attack, and verification tasks in an unknown region for targets that move in real time. These authors also address the problem of obstacle avoidance. Jin et al. [10] propose a probabilistic task allocation scheme for the scenario presented in [5, 6].

In most of the algorithms presented in the papers cited above, global communication between agents is assumed. So, information obtained by an agent is assumed to be communicated to all the other agents. However, the task allocation decision algorithm is autonomously executed by the agents. Sujit et al. [11] present a team theoretic approach that allows the UAVs to perform decision-making independently when there is no communication between UAVs. The UAVs can only sense the location of their neighbours and targets. The task allocation is performed based on a linear programming formulation, where the cost associated with each target is estimated with some probability. Rajnarayan and Ghose [12] pose a multiple agent search problem as a problem in the team theory framework and propose optimal solution to the problem of determination of search regions.

Recently, market based approaches have shown some considerable increase in performance of task allocation strategies for multiple agent applications. Dias and Stenz [13, 14, 15] introduce a novel approach for coordinating robots based on the free market architecture in economics. The approach defines revenues and cost functions across the possible plans for executing a specified task. The task is accomplished by decomposing it into sub-tasks and allowing the robots to bid and negotiate to carry out these sub-tasks. Gerkey and Mataric [16] use an auction mechanism for multi-robot coordination. Although there are many cooperative strategy algorithms available for multiple agents, very little attention has been paid to the role of communication which plays a vital role in the development and performance of a search strategy. Usually, the communication between agents is quantified by the amount of information that flows between them. The communication and computational complexity involved in multi-robot task allocation are analyzed in [17]. Mataric et al. [18] develop various task allocation strategies and study their performance on a multi robot application with sensor noise. The simulation study is compared with the results obtained using experiments. Gurfil [19] also uses auctions mechanism for task allocation among multiple UAVs performing search and destroy mission. Lagoudakis et al. [20] uses auction algorithm for assigning unexplored tasks to group of mobile robots. The paper also provides some theoretical bounds on the computational complexity of the proposed algorithm.

In this paper, we present a task allocation algorithm for multiple UAVs performing search and attack tasks in an unknown region using negotiation scheme for the scenario given in the next section. This is one of the very few applications available that exploits the use of negotiation for a network of UAVs involved in a practical problem of decision-making.

1.2 Problem Scenario

Consider a planar search space consisting of an unknown number of targets. The location of the targets are not known *a priori* to the UAVs. A search and destroy mission would be undertaken by sending a fleet of UAVs to search a region and destroy as many targets as possible within the flight endurance time of the UAVs. The task involved in such missions are search and attack. A search task is to search the environment for targets and, if found, engage a UAV to attack the target. The UAVs have to coordinate among each other so as to accomplish the mission efficiently and in minimum time. The UAVs have limited sensor and communication range. The UAVs can sense the exact position of all the targets and its value within its sensor range. The location of all the neighbouring UAVs within a UAV's sensor region are also assumed to be detected. The UAVs should perform task allocation so that as much of the mission as possible is completed. The task allocation mechanism we use is based on negotiation.

2. PROBLEM FORMULATION

Consider N UAVs/agents performing a search and destroy operation on a bounded region consisting of M targets whose exact positions are not known *a priori*. The basic problem of task allocation is to assign agent $A_i \in N$, to target $m_i \in M$, efficiently such that the mission is completed as fast as possible. The task allocation problem can be solved either using a centralized controller or a decentralized controller. In the former case each agent communicates the information it has to the centralized controller that solves a task allocation algorithm and assigns each agent to a particular task. However, implementing this task allocation strategy in real-time requires large communication overhead and may not be scalable with increase in number of agents and targets. Also, these strategies are not robust to failures. Hence, a decentralized task allocation strategy is necessary for implementing on a multi-agent system. The decentralized task allocation can be performed by any one agent in the multiple agent system. This agent receives the necessary information from all the agents and solves a task allocation algorithm and broadcasts the assignments to the agents. Another way of implementing a decentralized task allocation would be that, each agent broadcasts its information to all the other agents and hence each agent has the required information to solve the task allocation problem independently and assign a task for itself. The implementation of this task allocation

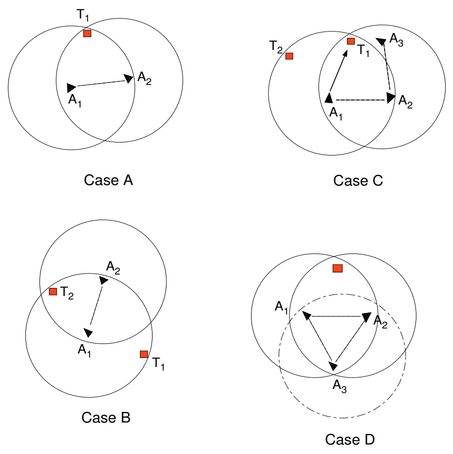

Figure 1: Some scenarios for decision-making

strategy requires large amount of communication among the agents.

The implementation of decentralized task allocation with finite communication range poses some challenging problems. For instance, consider Case A from Figure 1 where agent A$_1$ and A$_2$ have target T$_1$ in their sensor range and an allocation has to take place as to which agent should be assigned to the target. The task allocation can be done using a greedy strategy, in which case both the agents would move towards the same target which is not desirable. Another task allocation mechanism used in multi-robot literature is based on auctions. But in this case since the system is decentralized then each agent would become am auctioneer and hence both the agents auction the same target. A modification to the standard auction algorithm may eliminate some of the difficult issues, however this would complicate the decision-making rules for multiple agents using auction mechanism locally.

Consider Case B, wherein A$_1$ has T$_1$ and T$_2$ in its sensor range while A$_2$ has T$_2$ only. Auction requires broadcast of all the target and their associate costs. Resolving conflicts using auctions is a difficult task. In Case C, we can see that A$_1$ sees T$_1$ while A$_3$ is already on its way to attack T$_1$. So, A$_1$ wastes some resource in moving towards a target that is already assigned, Since the communication is limited it does not have access to the assignment of other agents. Instead of T$_1$ it could have attacked T$_2$. Here also greedy and auction algorithm would not yield good performance. In Case D, agent A$_3$ gets the auction information from A$_1$ and A$_2$ about T$_1$, now A$_3$ does not know to which agent it has to send the bid. These complications in using auctions for limited communication cases, give rise to the use of negotiation as a tool that can handle these situations efficiently. In cases A, A$_1$ and A$_2$ can negotiate which agent would be assigned to target T$_1$. While in case B, A$_1$ and A$_2$ can negotiate such that one agent attacks T$_1$ and the other moves towards T$_2$. In case C, A$_2$ can detect a conflict between A$_1$ and A$_3$ and send decisions such that A$_1$ or A$_3$ moves towards T$_1$. However in Case D, A$_3$ actually negotiates between A$_1$ and A$_2$ which are not neighbours and detects an early conflict and hence provides an efficient task allocation decision and has better fuel usage as only one agent can continue search to select other targets instead of moving to T$_1$.

However, the implementation of negotiation scheme involves designing of negotiation rules over which the process of decision-making takes place. In the next section we describe the negotiation scheme employed in decision-making.

At every time step each agent has to perform a task. The task can be search or attack a target. Each agent senses its environment consisting of other agents and targets. An agents' assignment for a task depends on four different situations. These situations are dependent on the availability of neighbouring agents and targets. The four situations, in which agent A$_i$ has to perform a task and make a decision are:

1. No targets and no neighbours

   Task: Search

   Decision: Continue to move in the same direction

2. No targets but has neighbours

   Task: Perform search or attack. The target information may be provided by the neighbouring agents.

   Decision: Acts as a negotiator for neighbouring agents

3. Targets are present but no neighbours

   Task: Attack

   Decision: Select a target that yields maximum value

4. Target as well as neighbours are present

   Task: Search or attack

   Decision: Negotiate with neighbours

Once an agent A$_i$ is present within a distance d from the target, we assume that the target is destroyed. The agents have to negotiate with its neighbouring agents for an efficient task allocation. The agents are not subjected to any turn radius constraints and hence can move in any direction. The agents have to maximize the number of targets destroyed in the search space by coordinating with its neighbouring agents through negotiation.

3. DECISION-MAKING

3.1 Negotiation as a tool to handle uncertainty in agent actions

In general, negotiation refers to the communication process that facilitates coordination and cooperation among a group of agents [22]. In multi-agent systems, its aim is to resolve problems related to resource allocation and task assignments between various agents in a decentralized setting.

Our approach is somewhat similar to Rubinstein's model of strategic negotiation [21] where agents make proposals that are either accepted or rejected by other agents; and whether an agent implements its proposal or not depends on what other agents do. However, our approach is different from Rubinstein's model on many counts due to the nature of the task allocation problem. Unlike most negotiation models we do not have a situation where each proposal is vetted by all the other agents. In fact, due to the connectivity restrictions we have a network of agents where an agent is not necessarily directly connected to all other agents. So, each agents decision is based on the response of

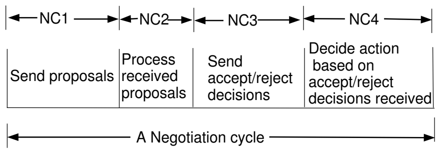
NC1
NC2
NC3
NC4
Send proposals
Process received proposals
Send accept/reject decisions
Decide action based on accept/reject decisions received
A Negotiation cycle

Figure 2: Negotiation cycle

only those agents that are connected to it. Moreover, unlike in Rubinstein's model, agents make simultaneous offers at pre-defined decision epochs and the actions are accordingly distributed between agents. Another way in which our model differs from Rubinstein's model is due to the fact that, in a task allocation problem the need for negotiation arises mainly because of lack of information about the action of other agents. So, the whole process of negotiation is geared towards determining the action of an agent in a coordinated autonomous fashion without assuming any kind of hierarchy or priority among agents.

A coordinated decision by an agent would be one that is not in conflict with the decision of its neighbors. There is no conflict except that which arises due to uncertainty of agent actions. For example, it occurs when more than one agent is planning to attack the same target, thus decreasing the effectiveness of the mission. Resolution of such conflicts can be effected either by

(i) Direct communication/negotiation as in the case when an Agent A_i and another agent A_j are within communication range.

(ii) Indirect negotiation when an Agent A_i and another agent A_j, $\mathrm{A}_j \notin \mathcal{N}(\mathrm{A}_i)$ want to attack the same target T, and A_i and A_j are connected through a sequence of communication links through other agents. For instance, they may be connected through a third agent A_k with $\mathrm{A}_j \in \mathcal{N}(\mathrm{A}_k))$ and $\mathrm{A}_k \in \mathcal{N}(\mathrm{A}_i)$.

In the first case, since A_j is within the communication range of A_i, it can exchange information with A_j and resolve the conflict. While, in the second case, A_i does not know about the existence of A_j, and so direct communication is not feasible. So, the intermediate agents are important in the negotiation process. In the negotiation scheme developed next, we will show that it is the neighboring agents who contribute to the decision-making of agent A_i.

3.2 Negotiation scheme

Each agent A_i performs the following actions during decision-making: (i) Sends/receives proposals (ii) Processes received proposals and sends Accept/Reject decisions to proposing agents (iii) Computes own route decision (iv) Implements decision. All these actions happen within each negotiation cycle. This is shown in Figure 2. Note that an agent A_i that has no targets will have the second segment only, while the agents that have targets as well as neighbouring agents will have all the four segments of decision-making.

The different segments of the negotiation cycle are described below:

3.2.1 Send/receive proposals (NC1)

Each agent evaluates the benefit associated with each target. Let $b_i(T_j)$ be the expected benefit A_i gets by attacking target T_j, which is given by

$$b_i(T_j) = V_{T_j} w_r \; - \; S_t \tag{1}$$

where, V_{T_j} = value of target T_j, w_r = the weight given to search task over the task of attacking a target, S_t = (time to reach the target T_j)/(total flight time). The benefit set $\mathcal{B}_i$ of A_i consists of benefits for all the tasks an agent has. Let $\mathcal{T}_i$ be the set of all targets. The benefit set for agent A_i is represented as:

$$\mathcal{B}_i = \{b_i(T_j) \mid T_j \in \mathcal{T}_j\} \tag{2}$$

Agent A_i chooses a target T_{S_i} for which A_i gets the maximum value, as

$$S_i \;=\; \arg\max_j \{b_i(T_j) \in \mathcal{B}_i\} \tag{3}$$

The proposal of agent A_i, sent to its neighboring agents, is of the form $Q_i = (\mathrm{A}_i, T_{S_i}, b_i(T_{S_i}))$, containing the proposer agent's identification, proposed target, and the value associated with T_{S_i}.

3.2.2 Processing received proposals (NC2) and send decisions (NC3)

Let $\mathcal{Q}_i$ be the set of proposals received by agent A_i from its neighbors A_j, including its own proposal.

$$\mathcal{Q}_i = \{(\mathrm{A}_j, C_{S_j}, \beta_j); L(\mathrm{A}_j) \in \mathcal{N}(L(\mathrm{A}_i), q_c)\}$$

Let T_k^i be a target that appears in at least one of the proposals received by A_i. That is, $T_k^i = T_{S_j}$ for some $Q_j \in \mathcal{Q}_i$. For each such T_k^i, define $\mathcal{A}(T_k^i)$ as the set of agents that have proposed T_k^i, and $\mathcal{B}(T_k^i)$ as the set of values associated with agents in $\mathcal{A}(T_k^i)$. So,

$$\begin{aligned} \mathcal{A}(T_k^i) &= \{\mathrm{A}_j \mid Q_j \in \mathcal{Q}_i, T_{S_j} = T_k^i\} \\ \mathcal{B}(T_k^i) &= \{b_i(T_j) \mid \mathrm{A}_j \in \mathcal{A}(T_k^i))\} \end{aligned} \tag{4}$$

Using the above two sets ($\mathcal{A}(T_k^i)$) and $\mathcal{B}(T_k^i)$, agent A_i sends *accept* or *reject* decision to its neighbors using the following rules:

Rule 1: An agent A_i sends *accept* to agent A_j, if

$$\mathcal{A}(T_k^i) = \{\mathrm{A}_j\} \tag{5}$$

That is, $\mathcal{A}(T_k^i)$ is a singleton containing only agent A_j (note that A_j could be A_i itself).

Rule 2: If $\mathcal{A}(T_k^i)$ is not a singleton then agent A_i sends *accept* to that agent in $\mathcal{A}(T_k^i)$ which has the maximum value by attacking target T_k^i and *reject* to all other agents in $\mathcal{A}(T_k^i)$. That is, *accept* is sent to $\mathrm{A}_{j'} \in \mathcal{A}(T_k^i)$ if,

$$j' = \arg\max_j \{b_i(T_j) \in \mathcal{B}(T_k^i)\} \tag{6}$$

Note that Rule 2 subsumes Rule 1. But they are stated separately for clarity. Again $\mathrm{A}_{j'}$ can be A_i itself.

Rule 3: An agent can send only one *accept* for one target. If there are more than one j' then the agent selects one of them.

Rule 4: For A_i to decide on its action at the current search step it has to get *accept* from all its neighboring agents to which it had sent its proposals.

Rule 1 implies that when an agents' proposal is not in conflict with other agents' proposals an *accept* can be sent without considering other agents' decision. When more than one agent proposes to attack T_k then there is a conflict between the proposing agents which A_i has to resolve. The conflict can be resolved by comparing the benfits' proposed by the agents. Agent A_i compares the $b_i(T_j)$ received for target T_k and sends *accept* decision to an agent A_k which has the highest $b_i(T_j)$ and *reject* decisions to the remaining agents. An agent A_i can receive a mix of *accept* and *reject* decisions from its neighbors. If we allow the agent to attack a target T_k, since it has got acceptance from some of the agents, this assignment would cause ineffective performance as multiple agents will get assigned to the same target. Hence, *Rule 4* guards against agents getting multiple assignment. The Rules 1-4 are the key to the negotiation scheme. While implementing *Rule 3*, we may encounter situations where more than one agent has the same $b_i(T_j)$, in which case we use a deadlock resolution scheme that resolves such deadlocks.

3.2.3 Computing route decision (NC4)

Agent A_i decides whether to implement or discard its proposed task based on the *accept* or *reject* decisions received from its neighbors. The agent implements its proposal if it receives *accept* decisions from all its neighbors and discards it if the agent receives a *reject* from even one of its neighbors. An agent that received a *reject* for its proposal from at least one neighbor will go on to the next negotiation cycle and this process will continue till it receives all *accept* decisions. An agent that has arrived at a decision (after receiving accept from all its neighbors) will not send any more proposals during subsequent negotiation cycles. The sequence of negotiation cycles will terminate automatically when all the cycles have converged to a decision. Later we will prove that only a finite number of negotiation cycles are necessary.

When an agent A_i receives reject for all its proposals, it adopts the search task.

3.3 Additional target information exchange

The agents that have received acceptance to their proposal may have other targets within its sensor range. An agent A_i can send this information to its neighbouring agents who can use it. The information that an agents sends is the target location and its value as perceived by A_i. This information will be more useful for those agents that may not have decided any targets but are neighbours of A_i. The target information broadcast by A_i can also be useful if all the proposals of agent $\mathrm{A}_j \in \mathcal{N}(\mathrm{A}_i)$ are rejected.

Once an agent receives the available targets from agent A_i, it can make assignment to any of the targets based on random number generation, greedy strategy, or start a negotiation with its neighbouring agents for obtaining an assignment. In this paper we use greedy strategy for simplicity.

3.4 Deadlock resolution mechanism

We define a deadlock, when an agent A_i is unable to decide to whom it has to send an acceptance. This situation can happen when more than one agent, with the same $b_i(T_j)$ value, seeks target T_j to attack. Since the $b_i(T_j)$ values are same, use of *Rule 2* is not possible and agent A_i cannot send acceptance to all the agents as that will violate *Rule 3*. There are two possible ways of resolving deadlock: loss information and token algorithm.

Loss information: In this scheme, agent A_i requests for more information from agents in $\mathcal{A}(T_k^i)$. This additional information will aid in effective decision-making. The additional information that an agent requests is the value of possible loss that each proposing agent suffers if it chooses the next best action instead of the proposed action. Let the new benefit vector for agent A_k be $\hat{\mathcal{B}}_k$ and the loss λ_k be evaluated using (7) as,

$$\begin{aligned}\hat{\mathcal{B}}_k &= \{\mathcal{B}_k \setminus b_i(C_{S_K})\} \\ \lambda_k &= \max \mathcal{B}_k - \max \hat{\mathcal{B}}_k \end{aligned} \tag{7}$$

where, $'\setminus'$ denotes set difference. When agent A_i requests for loss information, the loss λ_k is sent to agent A_i. Let Λ_i represent the set of loss information received from all the agents in $\mathcal{A}(T_k^i)$. An *accept* is sent to an agent A_j that satisfies the condition in (8) and *reject* is sent to the remaining agents.

$$\mathrm{A}_j = \arg\max_i(\Lambda_i) \tag{8}$$

Suppose there are multiple $b_i(T_j)$'s that are at the next highest level, then the same procedure needs to be repeated. Using the loss information does not guarantee that the deadlock will be resolved. This situation can arise when multiple agents have the same loss value. In that case, we use a token algorithm as given below.

Token Algorithm: Every agent A_i carries a unique token number K_i. Whenever the above situation (of the loss being equal) occurs wherein the agent is unable to decide to whom it has to send acceptance, the agent requests for token number of the agents $\mathrm{A}_k, \mathrm{A}_k \in \mathcal{A}(T_k^i)$. Let $\mathcal{K}_i$ be the set of token numbers received by agent A_i. Agent A_i compares these token numbers and chooses an agent A_j with the least token number.

The token number K_j is increased by a number $\hat{N}$, where $\hat{N}$ is an arbitrary large number greater than N. Agent A_j updates its token number to $K_j + \hat{N}$, after receiving the acceptance from agent A_i. This scheme ensures that an agent that has been selected earlier in this situation, will not be selected again in a similar situation if there is at least one other agent which has not been selected before.

Whenever a situation occurs when many agents are proposing for the same target, using the rules specified and the deadlock resolution mechanisms, it is guaranteed that there would be at least one agent that gets acceptance from all its neighboring agents.

3.5 Some theoretical results

THEOREM 1. *If more than one agent is proposing a target T_j, then at least one of the agents will receive all acceptances from its neighbors.*

Proof: Let $\mathcal{A}(T_j^i)$ be the set of agents proposing target T_j as their proposal. Then, by *Rule* 2, agent A_i sends an *accept* decision to agent A_j which has the maximum $b_i(T_j)$. If there are multiple agents with same $b_i(T_j)$ then A_i invokes the deadlock resolution mechanism by which one agent would receive an *accept.* □

THEOREM 2. *The negotiation terminates in a finite number of negotiation cycles.*

Proof: From Theorem 1 we observe that, at each negotiation cycle, at least one of the agents gets all *accept* and so decides

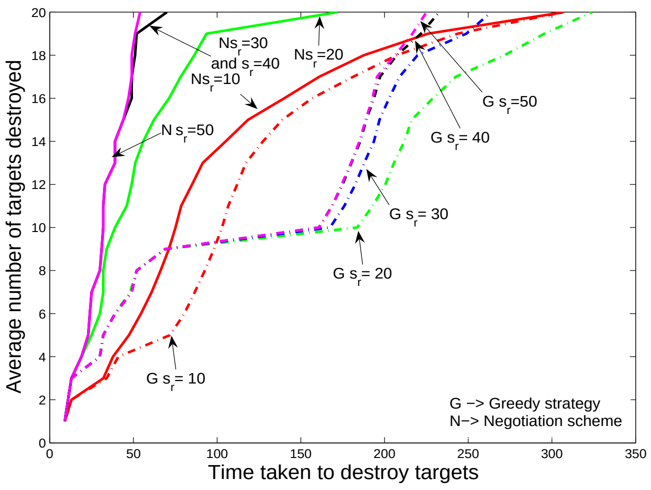

Figure 3: Average number of target hits for 100 different target positions

upon a target for its next step. Since there are a finite number of agents, in a finite number of negotiation cycles each agent would decide upon a target to attack. And if the target are not available then they choose search task. Hence, all the agents would decide upon a task in a finite number of negotiation cycles. The maximum number of negotiation cycles an agent can go through is N. $\square$

4. SIMULATION RESULTS

A simulation study is conducted on a battle field scenario of size 100×100. Through these simulations we show that the negotiation scheme performs better than greedy strategy in terms of average number of targets destroyed. The simulation is carried out using 7 UAVs for 100 different sets of target positions with each set having 20 targets. The *a priori* knowledge about number of targets present in the space and their initial positions are not available to the UAVs. We also study the performance of negotiation and greedy schemes for various sensor radius.

From Figure 3 we can see that the negotiation scheme outperforms the greedy strategy. The number of targets using negotiation scheme is higher and also the time taken to accomplish the mission is comparatively low. An expected result of increase in performance with increase in sensor range can be seen for the performance curves of negotiation scheme in the figure. However, this intuitive result is not true for greedy strategy for all the sensor radius.

As we can see from the figure the performance of greedy strategy with sensor radius $s_r = 10$, fares better than higher sensor radius $s_r = 20$ to $s_r = 50$. This is due to the fact with low sensor radius, the UAVs are unable to the sense the targets initially and hence move in the initial heading direction (spreading out). But, with higher sensor radius the agents are able to sense the target from their initial positions and hence all the UAVs move in the direction of sensed target as a swarm. Hence, the performance is worse when compared to lower sensor radius.

We carried out another set of simulations to study the performance of task allocation algorithm for different target

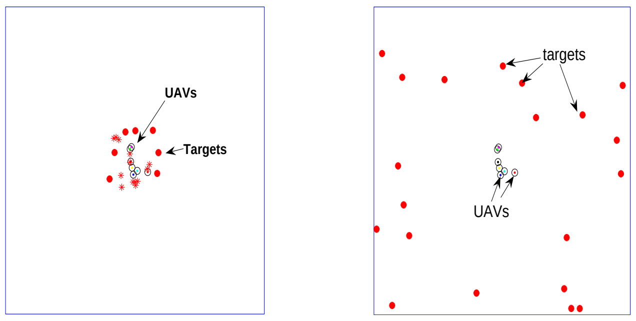

Figure 4: Battle field with 20 targets for proximity factors $\rho = 0.625$ and $\rho = 0.11$, while the sensor radius $s_r = 10$.

distributions on the search space. In order to conduct these experiments we define a proximity factor that determines the nature of the distribution or spread of targets in the search space. The proximity factor is defined as:

$$\rho = \frac{S_r}{\frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}} \tag{9}$$

where N is number of targets, (x_i, y_i) represents the position of the i^{th} target location, (x_c, y_c) the mean of all the target positions and S_r the sensor radius. Low proximity factor implies well separated targets compared to the sensor radius. While higher proximity factor ensures that the targets are placed very closely. Figures 4 show different target distributions in the search space.

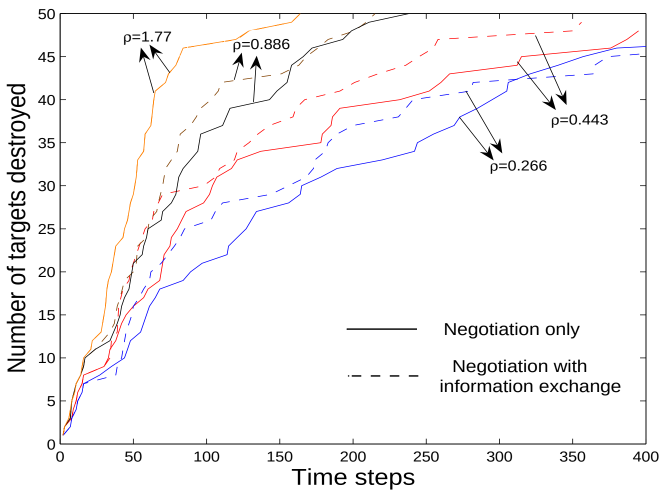

Figure 5: Number of targets destroyed for different proximity factors

The simulation is carried out using 7 UAVs for a search space consisting of 50 targets, with different proximity factors. Figure 5 shows the performance of negotiation and negotiation with target information based task allocation schemes. From the figure we can see that for lower proximity factors the number of targets destroyed are low as compared to the number of targets destroyed in the higher proximity factor case. When the proximity factor is small, the effect of target information sharing during decision making by the agents that have targets in their sensor range is significant. For $\rho = 0.266$, we can see from the figure that the performance of negotiation with target information based task allocation is better than that using negotiation only. Here, the target information broadcast plays a crucial role in enhancing the performance. Similar kind of effect can be seen for $\rho = 0.443$. However, for $\sigma = 0.886$, the negotiation based task allocation is better than that of with target information exchange. This is due to the fact that the additional information about distant targets makes the agent choose targets to attack rather than perform search in its own neighborhood. This causes it to miss nearer targets outside its sensor range. For $\rho = 1.77$, the performance is same for both the negotiation schemes. Since the proximity factor is high, all the agents can sense all the targets hence there is no improvement in performance with information exchange. It should be noted that the amount of information broadcasted also plays a crucial role in the performance of the task allocation. Hence, there is always a tarde off between how much of information should be broadcast and the performance.

5. CONCLUSIONS

In this paper, we demonstrate an efficient task allocation scheme using negotiation for multiple UAVs performing search and destroy mission on an unknown region. The performance achieved using negotiation based task allocation is compared with greedy strategy for various sensor ranges and it is found that negotiation based task allocation is significantly better. We also studied the effect of task allocation mechanism on the spread of targets in the search space. The results show that with lower proximity factor the number of targets destroyed is low while with higher proximity factor the number of targets destroyed is high. We also studied the performance of negotiation scheme when agent communicate additional information to its neighbouring agents about the availability of targets. It is found that with lower sensor radius this information yield better results.

6. REFERENCES

[1] K.E. Nygard, P.R. Chandler, and M. Pachter: Dynamic network flow optimization models for air vehicle resource allocation, *Proc. of the the American Control Conference*, Arlington, Texas, June 2001, pp. 1853-1858.

[2] C. Schumacher, P. Chandler, M. Pachter, and L.S. Pachter: UAV task assignment with timing constraints, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, August, Austin, Texas, 2003, AIAA 2003-5664.

[3] C. Schumacher, P.R. Chandler, S. J. Rasmussen, and D. Walker: Task allocation for wide area search munitions with variable path length, *Proc. of the American Control Conference*, June, Denver, Colorado, 2003, pp. 3472-3477.

[4] C. Schumacher, P. Chandler, and S. J. Rasmussen: Task allocation for wide area search munitions via iterative netowrk flow, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, California, August 2002, AIAA 2002-4586.

[5] P. Chandler, M. Pachter, S. J. Rasmussen, and C. Schumacher: Distributed control for multiple UAVs with strongle coupled tasks, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, August 2003, AIAA 2003-5799.

[6] P. Chandler, M. Pachter, S. J. Rasmussen, and C. Schumacher: Multiple task assignment for a UAV team, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, California, August 2002, AIAA 2002-4587.

[7] J.W. Mitchell, P. Chandler, M. Pachter, and S. J. Rasmussen: Communication delays in the cooperative control of wide area search munitions via iterative network flow, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, August 2003, AIAA 2003-5665.

[8] J.W. Curtis and R. Murphey: Simultaneaous area search and task assignment for a team of cooperative agents, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, August 2003, AIAA 2003-5584.

[9] D. Turra, L. Pollini, and M. Innocenti: Real-time unmanned vehicles task allocation with moving targets, *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Providence, Rhode Island, August 2004, AIAA 2004-5253.

[10] Y. Jin, A. A. Minai, M. M. Polycarpou: Cooperative real-time search and task allocation in UAV teams, *IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003, Vol. 1 , pp. 7 - 12.

[11] P.B. Sujit, A. Sinha, and D. Ghose: Multi-UAV task allocation using team theory, *Proc. of the IEEE Conference on Decision and Control*, Seville, Spain, December 2005.

[12] D.G. Rajnarayan and D. Ghose: Multiple agent team theoretic decision-making for searching unknown environments, *Proc. IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003, pp. 2543-2548.

[13] M.B. Dias and A. Stentz: A free market architecture for distributed control of a multirobot system, *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, Venice, Italy, July 2000, pp. 115-122.

[14] M.B. Dias, R.M. Zlot, N. Kalra, and A. Stentz: Market-based multirobot coordination: A survey and analysis, *Technical report CMU-RI-TR-05-13*, Robotics Institute, Carnegie Mellon University, April 2005.

[15] M.B. Dias and A. Stentz: Enhanced negotiation and opportunistic optimization for market-based multirobot coordination, *Techical Report CMU-RI -TR-02-18*, Robotics Institute, Carnegie Mellon University, August, 2002.

[16] B. Gerkey, and M.J. Mataric: Sold!: Auction methods for multi-robot control, *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, October 2002, pp. 758-768.

[17] B. Gerkey, and M.J. Mataric: A Formal Framework for the Study of Task Allocation in Multi-Robot Systems, *International Journal of Robotics Research*, Vol. 23, No.9, Sep 2004, pp. 939-954.

[18] M.J. Mataric, G.S. Sukhatme, and E.H. Stergaard: Multi-robot task allocation in uncertain environments, *Autonomous Robots*, Vol. 14, 2003, pp. 255263.

[19] P. Gurfil: Evaluating UAV flock mission performance using Dudeks taxonomy, *Proc. of the American Control Conference*, Portland, Oregon, June 2005, pp. 4679-4684.

[20] M. Lagoudakis, P. Keskinocak, A. Kleywegt, and S. Koenig: Auctions with performance guarantees for multi-robot task allocation, *Proc. of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, September 2004, pp. 1957-1962.

[21] A. Rubinstein: Perfect equilibrium in a bargaining model, *Econometrica*, Vol. 50, No. 1, 1982, pp. 97-109.

[22] S. Kraus: Automated negotiation and decision making in multiagent environments, *Multi-Agent Systems and Applications*, Springer LNAI 2086, (Eds.) M.Luck, V. Marik, O. Stepankova, and R. Trappl, 2001, pp. 150-172.