Distributed Scheduling Using Constraint Optimization and Multiagent Path Planning

Christopher T. Cannon[1], Robert N. Lass[1], Evan A. Sultanik[1], William C. Regli[1], David Šišlák[2], and Michal Pěchouček[2]

[1] Department of Computer Science, CoE, Drexel University
{ctc82,urlass,eas28,regli}@drexel.edu
[2] Agent Technology Center, FEE, Czech Technical University in Prague
{david.sislak,michal.pechoucek}@agents.felk.cvut.cz

Abstract. The goal of the distributed scheduling problem is to minimize the global cost of assigning n decentralized workers to m tasks at time points. This problem is further complicated in continuous environments because the entire state space cannot be searched. This paper presents a decentralized approach of dividing the distributed scheduling in continuous environments problem into two subproblems: distributed set covering and distributed multiagent path planning. First, we represent the problem of assigning workers (*i.e.*, covers) to tasks (*i.e.*, sets) as a Distributed Constraint Optimization Problem (DisCOP). Then, the DisCOP solver passes its solution to the distributed multiagent path planner who creates a conflict-free path for each worker to its assigned tasks. By first representing the problem as a DisCOP, it restricts the plan space to only a set of feasible plans. We apply this approach to the scenario of distributed scheduling for unmanned aerial vehicle surveillance. This approach is shown to relieve the strain on the distributed multiagent path planner by reducing the plan space more so than other distributed approaches.

1 Introduction

The goal of the distributed scheduling problem is to minimize the global cost of assigning n decentralized workers to m tasks at time points. This problem is further complicated in three-dimensional continuous environments because the physical state space cannot be completely searched.

As an example of a distributed scheduling problem, consider the real-world domain of distributed sensor networks [11, 15]. The distributed sensor network target tracking problem, shown in Figure 1, consists of four agents each equipped with Doppler radar sensors whose goal is to track moving targets. Each agent's sensor has three sectors that only one of which can be active at any given time. To accurately track a target, the target must lie within at least three agent's active sensors. In this example, the agents are the workers and the sensor sectors are the tasks. The agents schedule which sector of their sensor to activate by communicating over an ad-hoc network to the other agents. These types of distributed sensor problems have been previously solved using DisCOP [10, 25] and node localization [8, 9] algorithms.

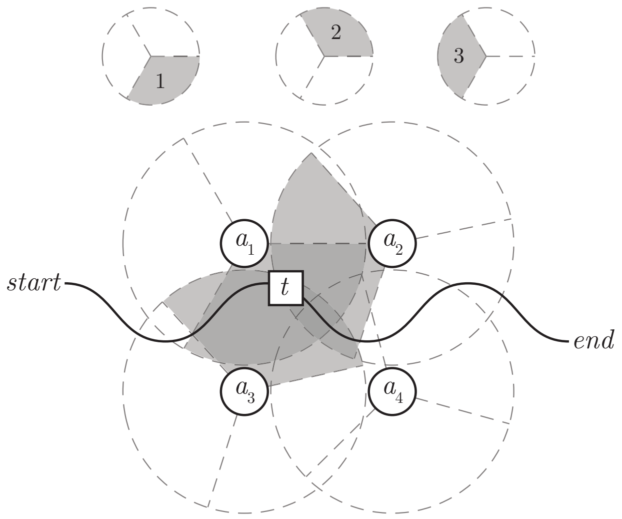


Fig. 1. A sensor network with four agents $(a_1, \ldots, a_4)$ tracking a moving target t.

This paper presents a decentralized approach of dividing the distributed scheduling in continuous environments problem into two subproblems: distributed set covering and distributed multiagent planning. To solve these subproblems, we utilize previous research in these areas and select the best performing techniques. First, we represent the distributed set covering problem as a Distributed Constraint Optimization Problem (DisCOP). Then, we represent the distributed multiagent planning problem as a distributed multiagent path planning problem. As shown in Figure 2, the approach consists of four steps:

1. The set covering problem elements are represented as workers and tasks (*e.g.*, a cover and set are equal to a worker and task, respectively) and costs for assignment are computed, which form the constraint graph of the DisCOP;
2. The DisCOP solver passes its solution to the distributed multiagent path planner;
3. The distributed multiagent path planner creates a conflict-free path for the workers to execute their assigned tasks; and
4. If an element of the problem changes (*e.g.*, a new worker, task, or cost), the distributed multiagent path planner passes the new set covering problem to the DisCOP solver and the process repeats.

The contributions of this paper are:

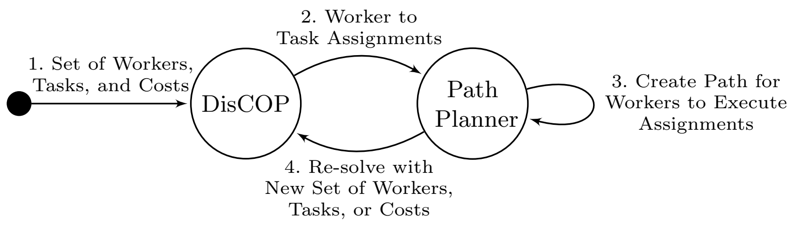


Fig. 2. Distributed scheduling with DisCOP and distributed multiagent path planning.

- A formalization of using DisCOP and multiagent planning for distributed scheduling in continuous environments;
- An example of mapping this formalization to the problem of unmanned aerial vehicle (UAV) surveillance; and
- A comparison of this approach to other fully distributed techniques.

The remainder of this paper is organized as follows: first we formalize the approach to solving the distributed scheduling problem. Then we map this formalization to the distributed scheduling problem of UAV surveillance. Using a flight simulation testbed, we analyze this approach against other fully distributed techniques. Finally, a discussion of future work and concluding remarks.

2 Formalization

This section presents a formalization of using DisCOP and multiagent planning to solve the distributed scheduling problem.

2.1 DisCOP

There are four components to a Distributed Constraint Optimization Problem (DisCOP)[3]: a set of *agents* $A = \{a_1, a_2, \ldots, a_n\}$, a set of *variables* that are to be assigned values by the agents $V = \{v_1, v_2, \ldots, v_{|V|}\}$, a set of *domains* that contain the values that may be assigned to said variables $\mathcal{D} = \{D_1, D_2, \ldots, D_{|V|}\}$, and a set of *constraints* over the variables' assignments. A constraint function for a pair of variables v_i, v_j is defined as $f_{ij} : D_i \times D_j \to |V|$. The objective is to have the agents assign values (taken from the domains) to their variables such that some metric over the resulting constraints' values is either minimized or maximized.

2.2 Multiagent Planning

There are seven components to the domain of a multiagent planning problem[4]: a set of *agents* $A = \{a_1, a_2, \ldots, a_n\}$, a set of *propositions* that are the agents beliefs

[3] Adapted from Modi *et al.* [16].
[4] Adapted from Bowling *et al.* [4].

$\Phi = \{\phi_1, \phi_2, \ldots, \phi_{|\Phi|}\}$, a set of *valid states* that are all possible combinations of said propositions $S = \{s_1, s_2, \ldots, s_{|S|}\}$ where $S \subseteq 2^{\Phi}$, a set of initial states $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{|\Theta|}\}$, a set of *actions* for $a_i \in A$, $\Gamma_i = \{\gamma_{i,1}, \gamma_{i,2}, \ldots, \gamma_{i,|\Gamma_i|}\}$, a transition relation of *effects* that are applied after said actions are executed, and a set of goal states $\Psi_i = \{\psi_1, \psi_2, \ldots, \psi_{|\Psi_i|}\}$.

2.3 DisCOP and Multiagent Planning Integration Formalization

Using the two previous formalisms, we formally show how the solution produced by the DisCOP solver prunes infeasible states (and associated actions) from the multiagent planning domain.

First, the DisCOP assigns intermediate goal states to agents: there is one DisCOP variable for each agent/state pair, the domain of which is a boolean indicator for whether or not that agent will set that state as a goal. Constraints in the DisCOP are domain specific, but generally dictate the cost of an agent pursuing a state (which may be based upon a heuristic). See Section 3.3 for an example of how costs are calculated for the UAV surveillance scenario. The solution to the DisCOP is then used to prune irrelevant states from the planning domain (*i.e.*, states that are unreachable).

We first define the function *associate* which prunes the multiagent planning problem's domain based on the DisCOP's domain. Formally,

$$\mathit{associate} : \Phi \times S \times \Gamma_i \times D_i \rightarrow D'_i \subseteq D_i.$$

With the *associate* function, we can define the pruned set of valid plan states. Formally,

$$S' = \{s \in S : \mathit{associate}(s, D_i) \neq \emptyset\}.$$

The function *relevant* maps the solution context of the DisCOP and the set of actions from the planning domain to a set of pruned actions. Formally,

$$\mathit{relevant} : \bigcup_{Q \in 2^V} \prod_{v_i \in Q} (\{v_i\} \times D_i) \times 2^{\Gamma_i} \rightarrow 2^{\Gamma_i},$$

with the mapping explicitly defined as,

$$\mathit{relevant}(t, \Gamma_i) \mapsto \{\gamma \in \Gamma_i : \langle \gamma, \textsc{True} \rangle \in t\}.$$

In summary, we prune the plan's domain, states, and actions based upon the DisCOP's task assignments. We first accomplish this with the *associate* function, which prunes the plan's domain to that of the DisCOP's domain to create a new plan domain. Then, we prune the plan states to only the states that are valid based on the new plan domain. Finally, with the *relevant* function, we prune the plan's actions based on the DisCOP's variables and domain.

3 Case Study: UAV Surveillance

As an application of this technique, we present the problem of multiple UAVs monitoring a set of targets. This section provides an informal and formal description of the problem and maps it to the DisCOP and multiagent planning formalization described earlier. We also describe the algorithms used for each solver and how they interact.

3.1 Scenario

Consider a set of four UAVs equipped with cameras (or any other type of sensor) that are monitoring a set of twelve enemy targets. The main goal of this scenario is to minimize the amount of time between a UAV surveilling a target. A concern of this scenario is that an enemy will capture a UAV, read its on-board data, and determine which targets are of most interest to its opposition. To negate this action, each UAVs target assignments only consist of a subset of the total targets to add an element of privacy to the locations of targets. A trusted centralized target assignment source is not feasible because assignments must be rapidly generated and a communications link to a centralized source in this scenario would likely be low bandwidth and therefore high latency. The four UAVs are equally divided into Group A and Group B, and the twelve targets are equally divided into Group A Exclusive, Group B Exclusive, and Shared. To prevent the disclosure of all targets if an enemy captures a UAV, each UAV group is only assigned to monitor its own exclusive group and the shared group of targets (*e.g.*, UAV Group A would monitor target Group A Exclusive and Shared).

3.2 Formalization

This problem is formally described as follows. Each UAV has a local agent $a \in A$, an entry position (where the UAV takes flight) $e \in E$, and an exit position (where the UAV lands) $x \in X$. Each UAV agent $a \in A$ surveils k targets $T = \{t_1, t_2, \ldots, t_k\}$ throughout a period of time. Time is divided into q rounds $R = \{r_1, r_2, \ldots, r_q\}$. The size of each round $r \in R$ is denoted as δ_R. In the experiments conducted, the time quanta for δ_R is one minute. An agent a has covered target t if it is there for the entire duration of that round. If at least one agent a covers target t during round r that target-round pair $\langle t, r \rangle$ is assigned the value 1, otherwise 0. The domain for each agent a_i consists of all pairs of targets and rounds: $D_i = T \times R$.

The function

$$empty(t_i) = \sum_{j=1}^{q} (r_j - \ell_{i,j})/\delta_R$$

outputs the total amount of uncovered time for target t_i. In the equation, $\ell_{i,j}$ is the last visited time an aerial vehicle was covering t_i during round r_j. The goal of the scenario is to minimize the number of rounds that all targets remain uncovered based upon a threshold value. Formally:

Minimize

$$\max_{i=1 \ldots k, j=1 \ldots q} \ell_{i,j}$$

Subject to

$$\sum_{i=1}^{k} \mathit{empty}(t_i) \leq \mathit{Threshold\ Value}$$

3.3 UAV Scenario to DisCOP and Multiagent Planning Mapping

Mapping the UAV surveillance problem to the formalization of DisCOP and multiagent planning is as follows:

- **Agents:** Each UAV agent $a_i \in A$ corresponds to the DisCOP and multiagent planning agent $a_i \in A$.
- **Propositions:** Each proposition $\phi_i \in \Phi$ corresponds to a target $t \in T$.
- **States:** Each state $s_i \in S'$, where S' is the pruned set of valid states from the *associate* function, corresponds to every combination of targets who are in the domain D'_i with values $\textsc{True}(\mathit{Covered})$ or $\textsc{False}(\mathit{Not\ Covered})$.
- **Variables:** Each agent $a_i \in A$ has a set of variables $v_i \in V$ which contains a single variable v_j for each constrained target. These variables correspond to the DisCOP formalization of V and must be one of the valid combinations $s_i \in S'$.
- **Domain:** Every $D_i \in \mathcal{D}$ is the same for each variable $v_{i,j}$ is boolean, $D_i = \{\mathit{Covered}, \mathit{Not\ Covered}\}$.
- **Actions:** The actions $\gamma_{i,j} \in \Gamma_i$ correspond to the combinations of all possible sequences of targets selected based upon the $\mathit{relevant}(t, \Gamma_i)$ function which specifies all targets set to $\textsc{True}(\mathit{Covered})$.
- **Initial State:** The initial state set $\theta_i \in \Theta$ corresponds to the entry position set $e_i \in E$.
- **Goal State:** The goal state set $\psi_{i,j} \in \Psi_i$ corresponds to the exit position set $x_i \in X$.
- **Constraints:** The cost of each variable assignment, where v_i and v_k represent two agents, and t_j is a single target is defined by the function

$$\mathit{cost}(v_i, v_k, t_j) = \begin{cases} \mathit{Low\ Cost} & \text{if } v_{i,j} = v_{k,j} = \mathit{Covered}, \\ \mathit{High\ Cost} & \text{if } v_{i,j} = v_{k,j} = \mathit{Not\ Covered}, \\ \mathit{No\ Cost} & \text{if } v_{i,j} \neq v_{k,j}. \end{cases}$$

Each of the costs in the cost function have a value based upon the current environment. *Low Cost* is equal to the number of UAVs constrained with the target (*e.g.*, if five UAVs are constrained with target t_j, $\mathit{Low\ Cost} = \mathit{5}$). In this situation, multiple UAVs are monitoring the target, but ideally there should only be one UAV associated per target. *High Cost* is equal to twice the number of UAVs in the scenario (*e.g.*, if their are 10 UAVs in the scenario, $\mathit{High\ Cost} = \mathit{20}$). Lastly, *No Cost* is always equal to zero, as it is the ideal situation when only one UAV is monitoring a target.

3.4 Distributed Stochastic Search

The distributed stochastic search (DSA) family of algorithms, specifically DSA-B variant, is the DisCOP algorithm used in the UAV experiments. As shown by Zhang *et al.* [23], the DSA-B algorithm has the best balance of solution quality, communication cost, and any-time properties compared to the other variants in the DSA family. The DSA-B algorithm first selects random values for variables in its *context*. A context is an assignment of values to variables for a DisCOP, which essentially is a (possibly partial) solution to a problem. Next, it enters a loop which continues until a solution is requested or a terminating condition is met. In the experimentation, the terminating condition was met when an agent did not update its variable after a period of time elapsed. In the loop, variables that were updated with a new value get sent as messages to its neighbors. Then the agent collects the messages received from its neighbors containing their variable assignments. After collection, the agent stochastically decides whether or not to update its value based upon whether there is a conflict between its currently assigned value and the value it received from its neighbors.

There are two reasons for choosing DSA-B over other complete algorithms: computational and memory cost, and fault tolerance [16, 18, 5, 13]. Because of the real-time computing restraints of this scenario and the limited hardware that would be available on an actual UAV, minimizing the computational and memory cost is critical. At any time DSA-B can provide its current context solution. During experimentation, a solution was requested on average every thirty seconds. If a complete algorithm were used, this would probably not be enough time to produce the optimal solution. DSA-B is also fault tolerant in the sense that an agent can drop out at any time, and the algorithm will still terminate and generate solutions.

3.5 Accelerated A*

In current research, there is a lack of three-dimensional plus time path planning algorithms. The best existing solution is an A* based algorithm, Accelerated A* (AA*) [21], which is the multiagent path planning algorithm used for the UAV scenario. Due to its superb computational efficiency, AA* is used in decentralized manner. Each UAV performs individual planning that provides a smooth trajectory through the waypoints. The interaction between the individual plans is achieved either at the time of plan execution or by means of multiagent flight simulation. In either case, the AA* planner hosted by the individual UAV is executed each time a collision is detected and the trajectory needs to be re-planned. As opposed to A*, AA* significantly reduces the number of samples (states) in a three-dimensional continuous space through adaptive sampling. The samples are generated by parameterized path construction elements. The sampling size varies based on the closest distance to the nearest operating space boundary. The configurations of states are not known prior to the search because they are given by the initial configuration and the algorithm execution. The number of states is higher (denser state space) around the obstacles and lower (sparser state space) further away from the obstacles. The adaptive parametrization defines the sampling length of straight elements and sampling angles for turn elements.

3.6 Interaction

The interaction between the different components in the approach is shown in Figure 3. First, the targets are passed to the DisCOP agent as variables. Using the DSA-B algorithm, the DisCOP agent determines its assigned targets. Before the targets are passed to AA*, they need to be ordered in the shortest path possible. One can generalize finding the shortest path between a set of targets problem as having a complete graph $G = (V, E)$, $V = \{v_1, v_2, \ldots, v_n\}$ that is isometrically embeddable in the Euclidean plane. In graph G the goal is to find the Hamiltonian path where v_1 is a known vertex, this is also known as the Traveling Salesman Problem (TSP), which is **NP**-COMPLETE [2]. We use the 2-Opt exchange algorithm [19] to determine an approximate shortest path solution, which it does by removing the crossing edges in a graph. After the 2-Opt exchange algorithm creates an ordered sequence of targets, they are passed to the local UAV AA* multiagent path planner, which outputs a smooth path for the UAV to follow. This process can be altered if the AA* multiagent path planner reports that it cannot generate a flight path based upon the targets assigned. If this event occurs, process repeats.

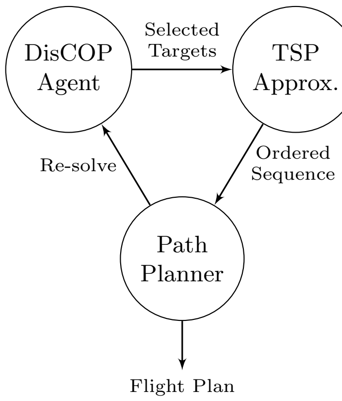


Fig. 3. State diagram of the local UAV agent view of the interaction between the DSA-B DisCOP agent, 2-Opt TSP approximation, and AA* multiagent path planner.

4 Experiments

This section provides empirical results of this paper's approach applied to the UAV surveillance scenario.

4.1 Related Work

Although there has been a significant amount of work in the area of UAV surveillance algorithms, most of this work relies upon a central decision maker and therefore is not fully distributed. For example, the Contract-Net Protocol [22] and other auction-based algorithms [12, 6] have been used to solve this problem. Although these solvers produce efficient results, they have no concept of privacy and allow one central authority to create plans for each UAV. Also, the Hungarian algorithm could be used to produce an optimal assignment if the problem reduced to a bipartite graph matching problem [17]. However, the bipartite graph matching problem assumes that a worker can only be assigned to one task, but in this problem a worker can be assigned to multiple tasks.

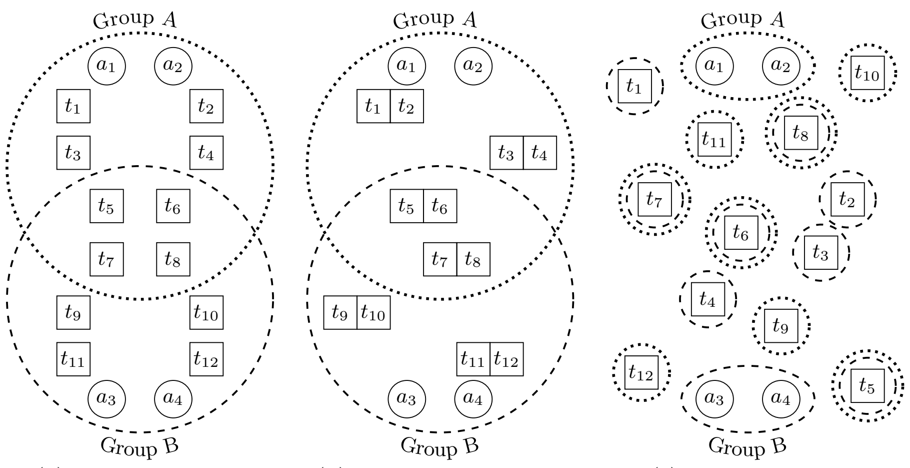
4.2 Experimental Parameters

As a comparison of our approach, we analyze its performance against two other distributed algorithms: *distributed greedy set cover* and *random selection*. The greedy set cover algorithm has been previously used over distributed nodes for a similar predator/prey problem [1]. In the algorithm, each UAV is assigned a unique identification number and sends out its current position and identification number to all other UAVs. Using this data, each UAV locally decides which targets it should monitor by selecting the lowest distance to each target that it was assigned to monitor. If a conflict exists where two or more UAVs are equal distance away from a target, then the UAV with the lowest identification number is assigned to that target. In the random selection algorithm, each UAV randomly selects whether it will monitor each target within its target group. For both of these algorithms, after a UAV completes its assigned targets, the algorithm is run again. This process continues until the scenario is terminated.

4.3 Experimental Setup

The AGENTFLY flight simulation testbed [20] was used for all the experimentation in this paper. Designed to test and compare collision avoidance algorithms in the domain of air-traffic control, AGENTFLY provides a three-dimensional space, support for limited communication, and real time 2D/3D visualization. Available air space can be restricted by terrain surface and no-flight zones.

To test the effectiveness of each algorithm, we mimic previous research in the area of measuring UAV surveillance techniques to define mathematically distinct target placements [7]. Three target arrangements were used: *uniform*, *two-cluster*, and *random*. In the uniform arrangement, targets are evenly placed throughout the environment. In the two-cluster arrangement, targets are paired and placed evenly around the environment. In the random arrangement, targets were randomly assigned locations. Each UAV is assigned to group A or group B to determine which targets they should monitor. For each of these arrangements, four targets were assigned to Group A Exclusive, four targets assigned to Group B Exclusive, and four targets assigned to Shared. These three target assignments are shown in Figure 4.



(a) Uniform targets. (b) Two-cluster targets. (c) Random targets.

Fig. 4. Scenarios including four UAVs ($a_1, \ldots, a_4$) and twelve targets ($t_1, \ldots, t_{12}$).

4.4 Metric Selection

To derive goal-based metrics for the scenario, we use the Goal Question Metric (GQM) approach by Basili *et al.* [3]. Metric selection using the GQM approach begins extracting the goals or high-level objectives from the specifications of the system. In the GQM approach, there are four parts to a goal: *purpose*, *issue*, *object*, and *viewpoint*. The *purpose* describes how the goal is effected. The *issue* is measurement of the event(s). The *object* is the item that the goal is centered around. The *viewpoint* is the perspective of the entity that is affected by the goal. An example given in [3] is to "Improve (*purpose*) the timeliness of (*issue*) change request processing (*object*) from the managers viewpoint (*viewpoint*)." Next, the evaluator identifies questions—usually with quantifiable answers—that when answered, decide whether or not the system meets the goal. Each goal may necessitate multiple questions. Finally, the metric is a set of data associated with the questions that can be subjective (depending on the point of view, such as ease of use of a user interface) or objective (which is independent of the point of view, such as program size).

Although one could identify many goals for this scenario, the most important goal is to minimize the number of rounds that targets remain unwatched. We apply the Goal Question Metric approach as follows:

Goal: Minimize the number of rounds (*purpose*) between surveillance (*issue*) of targets (*object*) from the view of each UAV (*viewpoint*).

- **Question 1:** What is the number of rounds between a UAV monitoring each target?
  - **Metric 1.1:** Measure the average gap in rounds between UAVs monitoring each target.

– **Question 2:** How well are targets globally assigned after each iteration of the algorithm?
  - **Metric 2.1:** Measure the average number of overlapping targets for each plan instance.
  - **Metric 2.2:** Measure the average number of excluded targets for each plan instance.

4.5 Analysis

The results for each algorithm and scenario are shown in Table 1 and Table 2. This paper's algorithm is referred to as DSA-B hereafter. Comparing only the average gap duration of unwatched target rounds, the distributed greedy set cover algorithm performs the best. However, on average there is only a 19% difference between it and DSA-B for all target types. The random algorithm performs comparably well on average, but also has a very high variance as one would expect. Comparing the results for overlapping and excluded assignments shows a true distinction between each algorithm and helps explain the results of Table 1. For each scenario, DSA-B produced plans on average with 38% fewer overlapping target assignments than distributed greedy set cover. This is important because overlapping target assignments mean that AA* must deconflict the airspace above the target for each UAV. Therefore these results show that selecting this paper's approach does in fact reduce the strain on the AA* multiagent path planner. However, the number of excluded targets explains why DSA-B does not dominate the performance of average gap duration of unwatched target rounds. Compared to the distributed greedy set cover and sometimes random algorithms, DSA-B excludes more targets on average. It is only logical to assume that the more plan instances which exclude targets will lead to an increased number of unwatched target rounds. The random algorithm again performs well, but its variance does not make it an attractive solution. Overall, the DSA-B algorithm has similar results to distributed greedy set cover of average gap duration of unwatched targets and consistently produces plans with few overlapping targets and therefore produces the best solution for this problem.

	Uniform			**Two-Cluster**			**Random**		
	Exclusive	Shared	All	Exclusive	Shared	All	Exclusive	Shared	All
DSA-B	3.8 ± 1.1	1.9 ± 0.2	3.2 ± 1.3	4.2 ± 1.9	2.3 ± 0.6	3.6 ± 1.8	4.9 ± 1.4	3.7 ± 1.5	4.5 ± 1.5
Greedy	$\mathbf{2.7 \pm 0.6}$	$\mathbf{1.7 \pm 0.2}$	$\mathbf{2.4 \pm 0.7}$	$\mathbf{2.4 \pm 0.6}$	$\mathbf{1.9 \pm 0.3}$	$\mathbf{2.2 \pm 0.5}$	$\mathbf{4.8 \pm 1.3}$	$\mathbf{3.6 \pm 1.0}$	$\mathbf{4.4 \pm 1.3}$
Random	4.2 ± 1.4	2.0 ± 0.5	3.5 ± 1.6	4.0 ± 1.6	2.5 ± 1.0	3.5 ± 1.6	9.2 ± 2.9	5.4 ± 2.4	7.9 ± 3.3

Table 1. Average gap duration of unwatched target rounds in minutes.

5 Conclusions and Future Work

This paper introduced a method of using DisCOP and distributed multiagent path planning to solve the distributed scheduling problem in continuous environments. We have formalized this method and applied it to the scenario of

	Uniform		**Two-Cluster**		**Random**	
	Overlap	Excluded	Overlap	Excluded	Overlap	Excluded
DSA-B	**4.63 ± 1.56**	2.13 ± 1.31	**4.82 ± 1.86**	1.44 ± 1.04	**5.13 ± 1.35**	0.87 ± 0.92
Greedy	6.87 ± 3.99	**0**	8.16 ± 3.55	**0**	8.46 ± 4.77	**0**
Random	4.82 ± 1.39	1.88 ± 0.88	4.94 ± 1.18	2.50 ± 1.34	5.62 ± 0.65	1.39 ± 0.87

Table 2. Average number of targets with overlapping and excluded assignments.

distributed scheduling for UAV surveillance. We have shown that this method does in fact reduce the amount of plan deconfliction by initially forming the problem as a DisCOP which restricts the plan space to only a set of feasible plans.

Future work will consist of experimenting with different DisCOP algorithms and altering the DisCOP constraints. An extension of DSA which remembers the best global solution explored [24] or an extension of the ADOPT algorithm altered to handle resource constraints [14] may produce solutions with less target exclusion which is ideal for this DisCOP application. The constraints of the DisCOP could be altered to also include the distance from the UAV's current position to its targets. This alteration would favor closer targets which may decrease the flight path distance, thereby decreasing plan execution time and the time between target surveillance.

References

1. M. Abramson, W. Chao, J. Macker, and R. Mittu. Coordination in Disaster Management and Response: A Unified Approach. *Lecture Notes in Computer Science*, 5043:162–175, 2008.
2. D.L. Applegate, R.E. Bixby, V. Chvatal, and W.J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.
3. V.R. Basili, G. Caldiera, and H.D. Rombach, editors. *Goal Question Metric Paradigm*, pages 528–532. Wiley & Sons, Inc., 1994.
4. M. Bowling, R. Jensen, and M. Veloso. A Formalization of Equilibria for Multiagent Planning. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence*, volume 18, pages 1460–1462, 2003.
5. A. Chechetka and K. Sycara. No-commitment branch and bound search for distributed constraint optimization. In *Proc. of the 5th Intl. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 1427–1429, 2006.
6. S.E. Conry, K. Kuwabara, V.R. Lesser, and R.A. Meyer. Multistage negotiation for distributed constraint satisfaction. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6):1462–1477, 1991.
7. M. Freed, R. Harris, and M. Shafto. Measuring autonomous UAV surveillance. In *Proc. of the Performance Metrics for Intelligent Systems Workshop*, Gaithersburg, Maryland, August 2004.
8. A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem. Distributed online localization in sensor networks using a moving target. In *Proc. of the 3rd Intl. Symposium on Information Processing in Sensor Networks*, pages 61–70, New York, NY, USA, 2004. ACM.
9. T. He, C. Huang, B.M. Blum, J.A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proc. of the 9th Annual*

Intl. Conf. on Mobile Computing and Networking, pages 81–95, New York, NY, USA, 2003. ACM.

10. M. Jain, M. Taylor, M. Tambe, and M. Yokoo. DCOPs meet the real world: Exploring unknown reward matricies with applications to mobile sensor networks. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence*, pages 181–186, 2009.
11. B. Laddaga, J. Sztipanovits, and V. Raghavan. Defense advanced research projects agency autonomous negotiating teams program. `http://people.csail.mit.edu/rladdaga/ants/`.
12. T. Lemaire, R. Alami, and S. Lacroix. A distributed tasks allocation scheme in multi-UAV context. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, volume 4, 2004.
13. R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proc. of the 3rd Intl. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 438–445, 2004.
14. T. Matsui, H. Matsuo, M. Silaghi, K. Hirayama, and M. Yokoo. Resource constrained distributed constraint optimization with virtual variables. In *Proc. of the 23rd Conf. on Artificial Intelligence*, pages 120–125, 2008.
15. P.J. Modi, H. Jung, M. Tambe, W.M. Shen, and S. Kulkarni. A dynamic distributed constraint satisfaction approach to resource allocation. *Lecture Notes in Computer Science*, 2239:685–700, 2001.
16. P.J. Modi, W.M. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.
17. C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity.* Dover Publications, 1998.
18. A. Petcu and B. Faltings. A distributed, complete method for multi-agent constraint optimization. In *Proc. of the 5th Intl. Workshop on Distributed Constraint Reasoning*, Toronto, Canada, September 2004.
19. G. Reinelt. The traveling salesman: Computational solutions for TSP applications. *Lecture Notes in Computer Science*, 840, 1994.
20. D. Šišlák, P. Volf, S. Kopřiva, and M. Pěchouček. AGENTFLY: A multi-agent airspace test-bed. In *Proc. of 7th Intl. Conf. on Autonomous Agents and Multi-Agent Systems*, May 2008.
21. D. Šišlák, P. Volf, and M. Pěchouček. Flight trajectory path planning. In *Proc. of Intl. Scheduling and Planning Applications Workshop*, pages 76–83, September 2009.
22. R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, 29(12):1104–1113, 1980.
23. W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, pages 55–87, 2005.
24. R. Zivan. Anytime local search for distributed constraint optimization. In *Proc. of the 23rd Conf. on Artificial Intelligence*, pages 393–398, 2008.
25. R. Zivan, R. Glinton, and K. Sycara. Distributed constraint optimization for large teams of mobile sensing agents. In *Proc. of the Intl. Joint Conf. on Web Intelligence and Intelligent Agent Technology*, pages 347–354, 2009.