Cooperation between search and surveillance agents in DCOP_MST

Roie Zivan and Katia Sycara,
Robotics Institute,
Carnegie Mellon University,
5000 Forbes Avenue,
Pittsburgh, PA, 15213, USA,
{zivanr,katia}@cs.cmu.edu

Abstract. A team of mobile sensors working together towards a common goal of covering targets in some area, can be modeled by DCOP_MST. DCOP_MST, is a model based on distributed constraints optimization that enables representation of the dynamic elements in a mobile sensing agents team problem, e.g., environment changes, changes in the agents' tasks and technology failures. Agents in DCOP_MST, perform local DCOP algorithms, adjusted to this model, in order to adjust their deployment to the current state of the dynamic environment.
In our previous work on DCOP_MST we assumed complete and accurate knowledge on the location and importance of targets in the area. This information might not be available in realistic scenarios. In this paper we consider two teams of agents, (a) sensors with advanced mobility and sensing tasked to detect targets, and (b) sensors with limited mobility tasked with target surveillance.
We demonstrate how the DCOP_MST model can be used by these two teams to fulfill their sub-tasks, and how it can be used for cooperation among the two teams. Our experimental study demonstrates how an increased level of cooperation among the two teams improves the performance of both teams and improves the overall result on fulfilling the mutual coverage goal.

1 Introduction

Some of the most challenging applications of multi agent systems include a team of mobile agents with sensing abilities which are required to cover a given area to achieve a common goal. Various examples are a network of sensors tracking enemy targets, rescue teams searching for survivors and teams of unmanned vehicles (UVs) searching an unfamiliar terrain. These applications are often large and complicated and thus, it is reasonable to assume that the agents cooperating to achieve coverage in these applications, reside on mobile sensors of different types and are divided into different teams where each of them has its own subtask. These subtasks are derived from the general common goal.

In a previous paper [12] we presented DCOP_MST, a model based on distributed constraint optimization for representing and solving problems of teams of mobile sensing agents [12]. The DCOP_MST model allows the representation of multiple dynamic

elements, which the applications discussed above include, such as changes in the environment, changes caused by technology failures and changes caused by agents' movements. We demonstrated in [12] how the DCOP_MST and the novel algorithms we proposed for this model, are successful in maintaining high level coverage in dynamic environments. The innovation of the DCOP_MST model is its ability to maintain dynamic domains and dynamic sets of neighbors. In addition, the environment is modeled as a function (environment requirement function, ER) that assigns to each point in the area its coverage importance, e.g., targets of interest are expected to have very high ER values.

However, in [12] we assumed that the ER function, which agents refer to and perform upon, is accurate and there is no uncertainty. This assumption is limiting when considering military and rescue applications. In this paper we extend the work on DCOP_MST. Here, the model is used to represent two teams of agents which reside on mobile sensors. The mutual goal of these teams performing in the same area, is surveillance coverage of all targets in the area according to their importance. The first team includes agents with advanced mobility and accurate sensing technology. The sub-task of this team of agents is to detect the targets and inform the surveillance agents (agents from the other team) of their location. The second team is the team of agents we have presented [12], which in this paper we refer to as "surveillance agents". Their task is to maintain coverage according to the targets' importance [12].

We demonstrate in this paper that the differences between the sub-tasks of the two teams require different search strategies. For the surveillance team, in [12] we found that the most successful algorithm was MGM enhanced with exploration methods which maintain reasonable coverage (prevent from abandoning covered targets). The requirement of maintaining long term coverage encouraged an algorithm in which the coordination between agents is high (an agent moves only if her neighbors do not) and monitored exploration level. On the other hand, search agents are expected to benefit from high level of exploration which will allow them to reach the entire area in minimal time.

To accomplish the different requirements of the search team's task, we designed a new algorithm based on DSA [9]. The agents use a second ER map/function which initially includes only vague probabilities on the location of the targets. The agents update this function by reducing the probability of areas which they recently visited. Agents select their next position according to this function. Our experiments show that in contrast to standard DSA [9], our algorithm does not suffer from thrashing as the probability for changing location increases.

Although the tasks of the two teams are different and require different algorithms, awareness of the global goal and the subtask of the other team can enable agents to cooperate with agents from the other team [3, 4]. We propose three levels of cooperation:

1. Agents from the search and detection team participate in the surveillance process of targets within their sensing range. This requires agents of the search and detection team to be aware of the location of the targets which were already found and to perform surveillance as long as they are in range.
2. Agents from the surveillance team inform the search and detection agents on elements in their sensor range which they suspect to be targets.

3. Search and detection agents which are covering a target, do not leave it until it is covered by surveillance agents.

.

In our experimental study we evaluated both the success of the sub-tasks of the two teams and the overall result when these different levels of cooperation are performed. Our empirical evaluation reveals that higher level of cooperation improves both.

The rest of this paper is organized as follows: After presenting the standard DCOP model in Section 2, we present DCOP_MST in Section 3. Section 4 presents the algorithm based on MGM performed by the surveillance agents. In Section 5, we present the extension to the model and the algorithm based on DSA performed by the search agents team. Section 6 discusses the different levels of cooperation we propose between the two sub-teams. Our empirical study is presented in Section 7 followed by our conclusions.

2 Distributed Constraint Optimization

A $DCOP$ is a tuple $< \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} >$. $\mathcal{A}$ is a finite set of agents $A_1, A_2, ..., A_n$. $\mathcal{X}$ is a finite set of variables X_1,X_2,...,X_m. Each variable is held by a single agent (an agent may hold more than one variable). $\mathcal{D}$ is a set of domains D_1, D_2,...,D_m. Each domain D_i contains the finite set of values which can be assigned to variable X_i. $\mathcal{R}$ is a set of relations (constraints). Each constraint $C \in \mathcal{R}$ defines a non-negative *cost* for every possible value combination of a set of variables, and is of the form $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary constraint* refers to exactly two variables and is of the form $C_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary DCOP* is a DCOP in which all constraints are binary. An *assignment* (or a label) is a pair including a variable, and a value from that variable's domain. A *partial assignment* (PA) is a set of assignments, in which each variable appears at most once. *vars(PA)* is the set of all variables that appear in PA, $vars(PA) = \{X_i \mid \exists a \in D_i \wedge (X_i, a) \in PA\}$. A constraint $C \in \mathcal{R}$ of the form $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$ is *applicable* to PA if $X_{i_1}, X_{i_2}, \ldots, X_{i_k} \in vars(PA)$. The *cost of a partial assignment* PA is the sum of all applicable constraints to PA over the assignments in PA. A *full assignment* is a partial assignment that includes all the variables ($vars(PA) = \mathcal{X}$). A *solution* is a full assignment of minimal cost.

3 DCOP_MST

DCOP_MST includes a number of concepts which are not used in the standard DCOP model. The first is the position of an agent. We denote the current position of agent A_i by cur_pos_i. The position of the agent is its current assignment but in DCOP_MST it is a physical position in the area (which can be represented by coordinates).

Second, we define two ranges which are essential in DCOP_MST. The first is the *Sensing Range*(SR) of agents. The sensing range is the effective coverage range of the agent, i.e., agent A_i can detect and cover all the targets that are within its sensing range from cur_pos_i. The *Mobility Range* (MR) is the range that an agent can move in a

single iteration of the algorithm. We denote the sensing range and the mobility range of agent A_i by SR_i and MR_i respectively.

After defining these concepts we can redefine the set D. A domain D_i of agent A_i includes all the alternative positions which are within MR_i from cur_pos_i. The resulting domain is a dynamic set [1].

For each agent A_i a credibility variable $Cred_i$ is defined. The credibility variable is a real positive number which is calculated by a reputation model.

We further define an environmental requirement function ER. This function expresses for each point in the area, the required joint credibility amount (the sum of the credibility variables) of agents with appropriate sensing range so that the given point can be adequately sensed (i.e. covered). Function Cur_DIFF calculates for each point in the area the difference between the current value of the ER function and the sum of the credibility values of the agents which are currently covering it. Formally, if we denote the set of agents within sensing range from point p by SR^p then:
$Cur\_DIFF(p) = ER(p) - \sum_{A_i \in SR^p} Cred_i$
The global goal of the agents in DCOP_MST is to cover all the area according to ER (i.e. to reduce the largest value of Cur_DIFF to zero). Since this goal cannot always be achieved, we define a more general goal which is to minimize the largest value of the Cur_DIFF function over all points in the area.

A set E includes all types of events. Each event can have an influence on the credibility of the agents involved in the event and/or on the function ER. A reputation model is used to define the influence for each event $e_j \in E$ on the credibility of the agents involved. An ordered set OE includes the events which occur according to their chronological order. Each member of the ordered set OE includes the type of event, the time of occurrence (iteration index), and its location (in case of an environmental event) or the agents involved (in case of an event which affects agents' credibility).

As in the standard DCOP model, each agent can send a message to each of the other agents. We assume that each agent is aware of the current position and credibility of each of the other agents. As in standard DCOPs neighboring agents are the agents that can be influenced by an assignment change (e.g. constrained agents). Thus, in DCOP_MST, two agents are considered neighbors, if after they both move towards each other, their sensing ranges overlap. Formally, the set of neighbors of agent A_i is denoted by cur_nei_i. An agent A_j is included in cur_nei_i iff the distance between cur_pos_i and cur_pos_j is less than $MR_i + MR_j + SR_i + SR_j$. Like in the case of the domains, since agents change their current position, the meaning of this definition is that the set of an agent's neighbors is dynamic.

4 MGM for DCOP_MST

The general design of the state of the art local search algorithms for $DCOPs$ is synchronous. In each step of the algorithm an agent sends its assignment to all its neighbors in the constraint network and receives the assignment of all its neighbors. The MGM

[1] An alternative definition would be that all the possible positions are included in an agent's domain but it only considers the ones with in its mobility range. However these two definitions are equivalent.

algorithm is a simpler version of the *DBA* algorithm [8, 9]. In every synchronous step, each agent sends its current value assignment to its neighbors and collects their current value assignments. After receiving the assignments of all its neighbors, the agent computes the maximal improvement (reduction in cost) to its local state it can achieve by replacing its assignment and sends this proposed reduction to its neighbors. After collecting the proposed reductions from its neighbors, an agent changes its assignment only if its proposed reduction is greater than the reductions proposed by all of its neighbors.

The adjustments required to apply MGM to solve DCOP_MSTs are as follow: First, as a self adjusting algorithm, the algorithm should run infinitely, i.e. after the algorithm converges to a solution it remains active in order to be sensitive to changes [2]. Second, the most delicate matter is the definition of the quality of each of the positions an agent can reach, so that it would serve the global goal. The global goal, as defined in Section 3 is to minimize the largest value of the Cur_DIFF function. The selection of the agents' positions must serve this goal. An immediate trivial choice would be a position which covers the point with the highest Cur_DIFF. However, in case there are a number of positions which enable coverage of this point, we would expect the agent to choose the most effective one, i.e., the position which enables coverage of additional points with a smaller Cur_DIFF. Therefore, an agent selects its position according to the following recursive method (its code is presented in Figure 1, method *select_pos*):

1. Each time the recursive method is called it is given a set of possible positions and a function that defines a value to each point in the sensing range of all the possible positions. In the first call, the set will include all the positions within the agent's mobility range MR_{self} and the function $Temp\_DIFF$ which is the current difference function without the current coverage of the agent performing the calculation (A_{self}). Formally, $Temp\_DIFF$ is defined as follow:
   For each point not currently covered by A_{self},
   $Temp\_DIFF = CUR\_DIFF$.
   For each point currently covered by A_{self},
   $Temp\_DIFF = CUR\_DIFF + Cred_{self}$.
2. Next, a set ($target\_set$) which holds the points with the largest function value in the sensing range of all of the agent's possible positions is generated.
3. Two termination conditions are checked:
   (a) If there is only one possible position, then it is selected.
   (b) If the largest function value is equal to zero (i.e. the $target\_set$ is empty). In this case any possible position can be selected.
4. If none of the termination conditions is met, the agent recalls the recursive method. The new set of possible positions which is passed in the recursive call includes all the positions in the current set of possible positions which are within the sensing range of all points in the target set. The function that is passed to the recursive method is the current function, only without the values of the points in the area which is within the sensing range of all positions in the new generated possible positions set. In other words, only areas which are not covered by the agent from each of the possible positions need to be considered.

MGM_MST

1. $cur\_pos \leftarrow$ Selected_init_pos()
2. **while** (true)
3. send cur_pos to each $A_i \in cur\_nei_{self}$
4. collect positions of each $A_i \in cur\_nei_{self}$
5. $LR \leftarrow$ **BestPossibleLocalReduction**()
6. Send LR to each $A_i \in cur\_nei_{self}$
7. Collect LRs from each $A_i \in cur\_nei_{self}$
8. **if** ($LR > 0$)
9. **if** ($LR > LRs$ of each $A_i \in cur\_nei_{self}$ (ties broken using indexes))
10. $cur\_pos \leftarrow$ the position that gives LR

BestPossibleLocalReduction()

11. $possible\_pos \leftarrow$ positions within MR_{self} from cur_pos
12. $Temp\_Diff \leftarrow Cur\_Diff \setminus self\_coverage$
13. $new\_pos \leftarrow$ **select_pos**($possible\_pos$, $Temp\_Diff$)
14. $cur\_cov \leftarrow$ highest $Temp\_Diff$ among points within SR_{self} from cur_pos and not within SR_{self} from new_pos
15. $new\_cov \leftarrow$ highest $Temp\_Diff$ among points not within SR_{self} from cur_pos and within SR_{self} from new_pos
16. return $\min(cur\_cov - new\_cov, Cred_{self})$

select_pos(pos_set , $func$)

17. **if** ($\|pos\_set\|$ = 1)
18. return pos_set.content
19. $target_set \leftarrow$points within SR_{self} from some $pos \in pos\_set$ with largest $func$ value (must be larger than zero)
20. if ($target\_set$ is empty)
21. return some $pos \in pos\_set$
22. **if** (no $pos \in pos\_set$ is within SR_{self} from all the points in $target\_set$)
23. $target\_set \leftarrow$ largest subset of $target\_set$ within SR_{self} from some $pos \in pos\_set$
24. $possible\_pos \leftarrow$ all positions in pos_set which are within SR_{self} from all points in $target\_set$
25. $intersect\_area \leftarrow$ area within SR_{self} from all $pos \in possible\_pos$
26. $new\_func \leftarrow func \setminus func.intersect\_area$
27. return select_pos($possible\_pos$, new_func)

Fig. 1. MGM_MST.

Figure 1 presents the code of the MGM_MST algorithm. The main loop of the algorithm remains almost unchanged from standard MGM [6, 7] (the standard algorithm was left out for lack of space). The agents send their assignments (current positions) to

the agents which are currently their neighbors. Notice that according to the assumptions in Section 3 the agent sends the accurate position [2].

Method **BestPossibleLocalReduction** calls method **select_pos** to find the best alternative position. After it is found, the method returns the improvement that would be achieved by changing to the selected alternative position. This improvement (or "reduction") is the difference between the highest Cur_Diff values, not including the credibility variable of A_{self} ($Temp\_DIFF$), which are covered by the agent when it is located in one of the two positions (the current and the new) and uncovered when it is located in the other (lines 13 - 15). The possible improvement can't be larger than the agent's credibility variable, $Cred_{self}$, since that is the agent's maximal contribution to the coverage of any point in the area (line 16).

Method **select_pos** is a recursive method that is first called with the set of all positions within the mobility range of the agent and function $Temp\_DIFF$. First, the two termination conditions (as described above) are checked (lines 17 - 21). In the process, a set called $target\_set$ is generated which includes all the points with the (same) highest value of the received function $func$, which are covered from at least one of the positions in the received set of possible positions (line 19). If all points in $target\_set$ cannot be covered from a single position, then only the largest subset of $target\_set$ which can be covered from a single position is left in $target\_set$ (lines 22,23). Next, a set is generated which includes all the positions in pos_set which enable coverage of all the points in $target\_set$ (line 24). In addition, a new function is generated which is equal to $func$ except for the values of the points in the area which is covered from all the positions in the new generated set ($intersection\_area$) (line 25,26). Finlay, the recursive method is called with the new generated set and function.

4.1 Exploration methods

Classic local search combines exploitation methods in order to converge to local optima and exploration methods in order to escape them [10]. The MGM_MST algorithm is strictly exploitive (monotone). While it benefits from quick convergence and avoids costly moves by the sensors, once a target is beyond the agent's range it remains uncovered. Algorithms which implement exploration methods were proposed for standard DCOPs [6, 9, 7, 11]. However, some of the methods which are most effective in standard DCOP are not expected to be effective for DCOP_MST [12].

In order to explore the area for new targets while maintaining coverage of targets which were previously detected we proposed two simple but powerful exploration methods which can be combined with the MGM_MST algorithm. These two methods change the parameters of the algorithm temporarily in order to escape local minima. This approach was found successful for local search in DisCSPs [1].

1. MGM_WR_MST simply allows an agent to consider points within a larger (double) range than their MR for a small number of iterations (WR represents Wide Range).

[2] This is a reasonable assumption considering that GPSs are used. We assume that the technology allows an agent to detect the agents which their ranges overlap with its own as defined in Section 3 and update its set of current neighbors. If not, agents would need to inform all other agents when they change position so they can update their set of neighbors accordingly.

This method assumes that a wider range is possible even though it is slower. Therefore the agents consider a wider range only in a small percentage of the algorithm's iterations which repeat periodically (in our experiments for example, we allowed two iterations of a wider, double, range every ten regular iterations).

2. The MGM_RC_MST algorithm allows agents in some iterations to move to a position which results in an increase of the Cur_Diff function up to some number c. More specifically, line 8 of the algorithm is changed in these iterations to:
   8. **if** $(LR + c > 0)$
   Again, this reduced condition (RC) is only temporary and is applied periodically. This would mean that for a small number of iterations the importance (coverage requirement) of targets in the area is reduced.

In both of these methods, agents are not expected to leave targets with high importance in order to search for new targets. In MGM_WR_MST it is obvious since like in the case of MGM_MST, only moves which result with a gain are performed. In the case of MGM_RC_MST, the c parameter defines the reduced importance of the targets which are already covered. Thus, c is a bound on the increase to the Cur_Diff function that the method can create.

In [12], the exploration methods described above were compared with standard MGM and two classic exploration methods adjusted to DCOP_MST, DSA and DBA.

5 Search and Detection team in DCOP_MST

The first step in including a search and detection team in DCOP_MST is relaxing the assumption that there exists an ER function which includes the accurate importance of every point in the area. Instead, we assume that the function initially includes some distribution which reflects the probability of the existence of targets in the area. This assumption makes the model compatible for any level of uncertainty from complete entropy (as in [5]) to complete knowledge (as in [12]).

This initial ER function is copied to another (initially identical) function we refer to as the *search map* (SM). Search agents use the search map SM when they decide on their path in order to detect targets and they update the ER function with the targets they find. The surveillance agents use only the ER function, in the same way as described in Section 4. Thus, the ER function is the device used for communication between the two teams.

The search agents use the SM function to communicate to each other where they have recently visited and therefore the probability of the existence of a new target is low. This is done as follow:

1. The base value of a point in the SM function, $Base_p$, is equal to the value in the initial ER function. Thus initially, all points in the SM function are equal to their base value.
2. A search agent sa located in some point p at iteration t, causes a decrease in the value of all points p' within the sensor range of p. The new value of these points is $SM(p', t) = Base_{'p} - Cred_{sa}$

DSA_SAT

1. $cur\_pos_{self} \leftarrow$ **select_init_pos()**
2. **while** (true)
3. **foreach** (point p within SR from cur_pos_{self})
4. $SM(p) \leftarrow Base_p - Cred_{self}$
5. $ER(p) \leftarrow$ **importance**(p) [3]
6. $next\_pos \leftarrow$ **get_best_pos()**
7. $rand \leftarrow random([0,1])$
8. **if** ($rand < prob$)
9. $cur\_pos_{self} \leftarrow next\_pos$

get_best_pos()

10. $possible\_pos \leftarrow$ positions within MR_{self} from cur_pos
11. $max\_val \leftarrow 0$
12. **foreach** $pos \in possible\_pos$
13. $psr \leftarrow p$ within SR from pos
14. **if** ($\sum_{p \in psr} SM(p) < max\_val$)
15. $next\_pos \leftarrow pos$
16. $max\_val \leftarrow \sum_{p \in psr} SM(p)$

Fig. 2. DSA_SAT.

3. At each iteration t in which there is no search agent in sensing range from point p, the SM value of p is incremented as follow:
   $SM(p,t) = min\{SM(p,t-1)+1, Base_p\}$.

Figure 2 presents the adjusted DSA algorithm for a team of search agents in DCOP_MST, DSA_SAT. In each iteration of the algorithm the SM value of all points within sensor range of the agent are set to a lower value according to the agent's credibility (lines 3,4). In addition, the agent updates the ER function with the true importance of the points in its sensing range. Then, the agent selects the best position it can move to by calling function **get_best_pos()**. The agent moves to this position in probability $prob$ (lines 7,8).

Function **get_best_pos()** selects the point within mobility range, which the sum of the SM values of points within sensing range from it, is maximal.

6 Cooperation across teams

In the previous sections we described two teams performing in the same area, each with its own task and means for communication between them via the ER function. However, it is reasonable to assume that cooperation among agents from the two teams can lead to better results for the following reasons:

1. Although each team of agents has its own task, they are both working towards a common goal.

2. Agents' efficiency depends on their location. Therefore, it may be the case that an agent from one team is in a position that allows her to serve the task of the other team best.

Common practice in multi agent systems include hierarchical plan structures that allow agents to assist others when working towards a common goal [3, 4]. Specifically to the applications at hand, we assume that search agents have superior technology and therefore they can perform surveillance with relatively high credibility while surveillance agents cannot define the importance of a target. Thus, we describe the following possible collaborations among the two teams:

1. Search support (SS): search agents take an active role in the surveillance of targets within their sensing range.
2. Alert: agents from the surveillance team increase the value of points in the search map where they suspect their might be a target. Thus, search agents are encouraged to search at these locations.
3. Avoid Abandoning (AA): search agents do not move to a new location when they are covering targets which are not reasonably covered by surveillance agents, i.e., search agents which locate a target wait for it to be covered by surveillance agents before they continue their search.

All three modes of cooperation described above require agents to be aware of the task of the other team. The alert and AA modes further require that agents communicate with agents in the other team via either the ER or the SM functions.

7 Experimental evaluation

In order to evaluate the performance of the two teams of agents and the effect of the proposed collaborations, we use the same simulator used in [12]. The problem simulated is of an area in which the possible positions are points on a grid. The size of the grid in all our experiments was 150 by 150. Each of the points in the area has an importance value between 0 and 100. The ER function initially had all points equal to 0. The credibility of search agents was set to 50 and the credibility of surveillance agents was set to 30. The sensing ranges of all agents were set to 10. The mobility range of search agents was set to 30 while the mobility range of surveillance agents was set to 10. The algorithm performed by the surveillance agents was MGM_WR_MST and the DSA_SAT was performed by the search agents. The area included 20 targets which are points with importance 100. We assume that these targets are revealed to search agents which are located within sensing range from them. Surveillance agents can perform their task only on targets which were detected by search agents. However, a target which has not yet been detected by a search agent raises the suspicion of a surveillance agent. All our experiments included 10 search agents and 60 surveillance agents. Each reported result is an average over 50 runs of the algorithm on different random generated problems. The random elements in each problem were the location of the targets and the initial location of the agents.

In the first set of experiments we evaluated the success of our proposed search agent algorithm DSA_SAT with respect to the value of the $prob$ parameter. Figure 3 presents

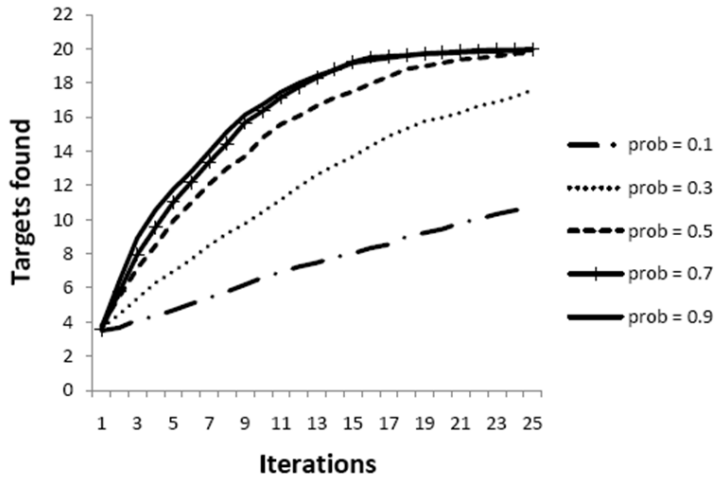

Fig. 3. Number of targets found by search agents using different levels of exploration.

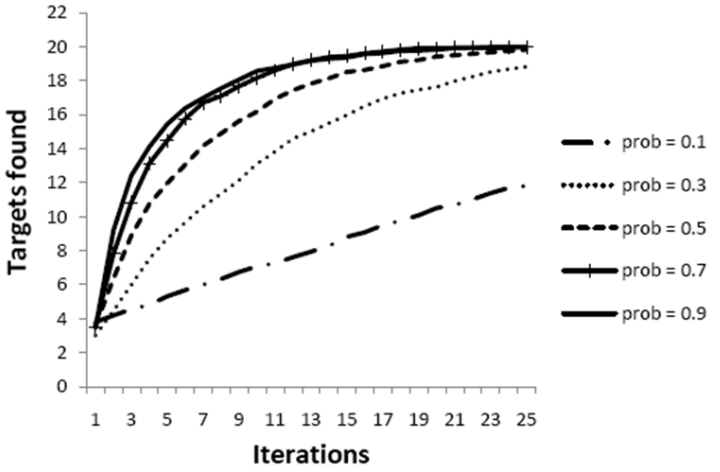

Fig. 4. Number of targets found by search agents using different levels of exploration (Alert).

the number of targets detected by the search agents as a function of the number of iterations performed since the search started. The different lines represent different $prob$ values used by the search agents in the DSA_SAT algorithm. It is clear that the algorithm is most successful for high values of $prob$. In contrast to standard DSA, high

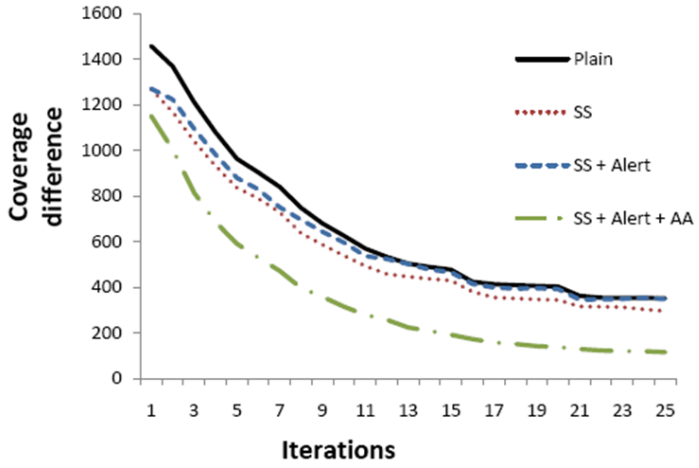

Fig. 5. Coverage difference for different levels of cooperation. (no search).

level of exploration does not cause thrashing. This is because the search agents are not required to converge to a solution like in standard DCOP but rather keep on searching for additional targets. However, for large $prob$ values (e.g., 0.7 and 0.9), there is no notable difference in performance. Figure 4 presents the same phenomenon when *Alert* cooperation mode is used. This figure demonstrates the benefit of using this mode of communication between the surveillance agents and the search agents for faster detection of targets.

Figure 5 presents the success of the two teams when performing together in order to reduce the difference of the sum of the ER values of all targets in the area and the credibility of agents which are performing surveillance in sensing range from the targets. In this set of experiments the initial ER function included all targets. Thus, there was no need to perform search. The results in Figure 5 demonstrate the effect of the different levels of communication on the performance of the surveillance team. It is clear from the result that when the search agents participate in the surveillance procedure (SS mode) but are not committed to it, the improvement in performance is minor. However, when the search agents are aware of the level of coverage on targets found and leave targets only if they are reasonably covered (AA mode), the performance in terms of surveillance substantially improves.

Figure 6 presents results of a complete experiment, i.e., target locations are not known in advance and the results are in terms of surveillance coverage (both sub-teams need to perform their sub-task). The results presented demonstrate how each additional level of cooperation among the two sub-teams improves the overall performance of the entire global team of sensing agents. The effect of the $Alert$ mode is more apparent in the first iterations when surveillance agents are waiting for targets to be discovered,

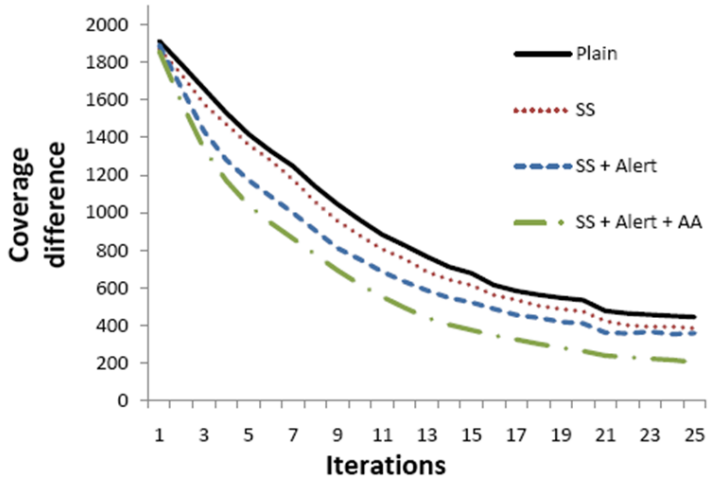

Fig. 6. Coverage difference for different levels of cooperation.

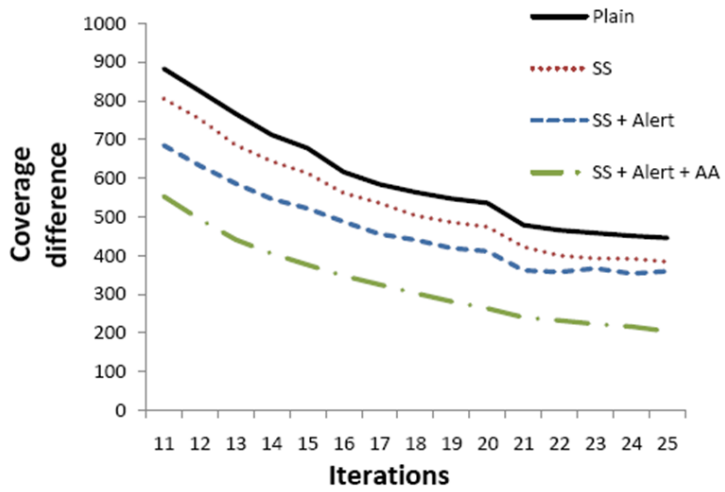

Fig. 7. Coverage difference for different levels of cooperation. (a closer look).

while the AA mode triggers the most substantial improvement in coverage. To emphasize the difference in performance, we focus on Figure 7 and see that after 25 iterations, in the highest level of cooperation, the coverage of the team is improved by a factor larger than two.

8 Summary and Conclusions

Recently, DCOP_MST, a new model based on DCOP for solving dynamic problems, was proposed. It represents a dynamic application of a team of mobile sensing agents which is expected to be robust to changes in the environment in which the sensors operate, changes in the teams tasks and technology failures. The early work, which introduced the DCOP_MST model [12], assumed that agents are aware of the location and importance of targets in the area.

In this work we adjusted the model to include two sub-teams of agents performing towards a common goal. Each team had its own sub-task which serves the common goal. The first team included agents with high mobility and accurate sensing technology. The subtask of this team was detection of targets. The second team's task was to perform surveillance on the targets which were detected by the first team. The search agents used an additional function, a *search map* (SM) to represent locations in which there is a higher probability to find targets. This SM was dynamically updated according to the search agents' movements. Thus, it was also used as a means of coordination among the search agents. The algorithm used by the search agents was based on DSA and our results demonstrated that it was most successful with high level of exploration and concurrency. The information regarding targets that were detected and their importance was transfered to the surveillance agents via the ER function.

Our results demonstrate the success of the DCOP_MST model in representing teams with different tasks. Furthermore, they demonstrate how increased levels of cooperation among the teams improve the performance of both sub-teams in fulfilling their sub-tasks and the result of the global team in achieving the common goal. Clearly, the flexibility of the model in representing and coping with dynamic elements, allows the teams to effectively handle additional information supplied to them by agents from the other team. The most successful results were achieved when all the proposed forms of cooperation were used.

Acknowledgment: This research has been supported by AFOSR FA9550-08-1-0356.

References

1. M. Basharu, I. Arana, and H. Ahriz. Solving coarse-grained discsps with multi-dispel and disbo-wd. In *IAT '07: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 335–341, Washington, DC, USA, 2007.
2. Z. Collin, R. Dechter, and S. Katz. Self-stabilizing distributed constraint satisfaction. *Chicago Journal of Theoretical Computer Science*, 5, 1999.
3. Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
4. Barbara J. Grosz and Sarit Kraus. The evolution of sharedplans. *Foundations and Theories of Rational Agency*, pages 227–262, 1999.
5. M. Jain, M. E. Taylor, M. Yokoo, and M. Tambe. Dcops meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. In *Proc. Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, Pasadena, CA, USA, July 2009.

6. R. T. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for dcop: A graphical-game-based approach. In *Proc. Parallel and Distributed Computing Systems PDCS)*, pages 432–439, September 2004.
7. J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *IJCAI*, Hyderabad, India, January 2007.
8. M. Yokoo. Algorithms for distributed constraint satisfaction problems: A review. *Autonomous Agents & Multi-Agent Sys.*, 3:198–212, 2000.
9. W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparishon and applications to constraints optimization problems in sensor networks. *Artificial Intelligence*, 161:1-2:55–88, January 2005.
10. Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.
11. R. Zivan. Anytime local search for distributed constraint optimization. In *AAAI*, pages 393–398, Chicago, IL, USA, 2008.
12. Roie Zivan, Robin Glinton, and Katia Sycara. Distributed constraint optimization for large teams of mobile sensing agents. In *International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 347–354, 2009.