Distributed Constraint Optimization for Large Teams of Mobile Sensing Agents

Roie Zivan, Robin Glinton and Katia Sycara,
Robotics Institute,
Carnegie Mellon University,
5000 Forbes Avenue,
Pittsburgh, PA, i5213, USA,
{zivanr,rglinton,katia}@cs.cmu.edu

Abstract

A team of mobile sensors can be used for coverage of targets in different environments. The dynamic nature of such an application requires the team of agents to adjust their locations with respect to changes which occur. The dynamic nature is caused by environment changes, changes in the agents' tasks and by technology failures.

A new model for representing problems of mobile sensor teams based on Distributed Constraint Optimization Problems (DCOP), is proposed. The proposed model, needs to handle a dynamic problem in which the alternative assignments for agents and set of neighbors, derive from their physical location which is dynamic. DCOP_MST enables representation of variant dynamic elements which a team of mobile sensing agents face. A reputation model is used to determine the credibility of agents. By representing the dynamic sensing coverage requirements in the same scale as the agents' credibility, the deployment of sensors in the area can be evaluated and adjusted with correspondence to dynamic changes. In order to solve a DCOP_MST, a local (incomplete) search algorithm (MGM_MST) based on the MGM algorithm is proposed and combined with various exploration methods. While existing exploration methods are evidently not effective in DCOP_MSTs, new exploration methods which are designed for these special applications are found to be successful in our experimental study.

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.

A common feature of the above applications is that the agents need to perform in a dynamic environment. There can be a number of reasons that cause the dynamic nature of such applications. One popular scenario which is considered in many studies is a change in the quality of information coming from agents/sensors [5] (throughout this paper we assume that each agent resides on a mobile sensor and use the terms *sensor* and *agent* interchangeably). Agents can pass incorrect reports as a result of technology failures or due to activities of malicious parties. However, there can be other dynamic elements in such an application. For example, intelligence reports that indicate a change of priorities in covering of different parts of the area, movements of units (which can be expected in military and rescue applications), and changes of tasks given by the operating authority.

Previous attempts to cope with the dynamic properties of mobile sensors have focused on a specific element of the problem. One example (mentioned above) is the detection of failing (or malicious) agents and avoidance of interaction with them [5]. Although in the case of failure of agents, detection is a first and important step towards the generation of a robust network, detection alone may not be enough. First, the indications for a failure might not be conclusive. Second, a change in the environment can cause the position of the agent to be no longer adequate. Thus, it would probably be more effective to relocate the agent than to avoid interactions with it. Third, in case of an agent's failure in an area with high importance, it is not enough to avoid interactions with it. The goal of the team is to maintain high level coverage on such delicate areas, thus, other functioning sensing agents should be moved in that direction.

Another example is the deployment of sensors in an area in order to achieve maximal coverage [8], [16]. In these studies agents make use of virtual potential fields in order to maintain an adequate distance from one another and thus, maximize the area they cover. In our work, a wider range of problems and tasks of mobile sensor teams is considered which include areas of high importance which require overlapping coverage, sensors with different level of credibility etc. We note that the max-coverage problem (e.g. cover the largest area) is a specific case of the problems which our proposed model applies to.

The present paper proposes a model and corresponding algorithms that makes a large team of mobile sensing agents robust to changes in the problem they face. The proposed model is an extension of the well known Distributed Constraint Optimization Problems (DCOP) model. In the proposed DCOP model for Mobile Sensing agent Teams (DCOP_MST), agents will adjust their position in order to adapt to the dynamically changing environment and the dynamic changes in the quality of information reported by sensors in the team.

Although DCOP is a general model that can be used to represent many real world problems as Meeting Scheduling [12], [6] and Supply Chain Management [2], its general design is static [10], [13], [15], [21], [7]. The study of DCOP algorithms in dynamic environments is in a preliminary stage [11], [9]. Our presented study is a step in the direction of the general and challenging goal to represent dynamic problems as DCOPs and design algorithms for solving them.

A mobile sensing agent team problem introduces a number of challenges which are not met by the standard DCOP model. The physical nature of the problem

requires that an agent will have a position. In addition, technology limitations result in bounded ranges of sensing and mobility. These ranges make the sets of relevant alternative positions and the set of agents which can be affected by an agent's movement, dependent on its current position. Therefore, the content of the domain and the set of neighbors of an agent may change when the agent moves. The DCOP_MST model allows agents to maintain dynamic sets of neighbors and dynamic domains.

One of the most challenging adaptations required in the design of DCOP_MST is the generation of a goal function which represents both the dynamic nature of the environment and the dynamic quality in agents' reports (e.g. their credibility). The credibility of agents is calculated using a *reputation model*. The use of a reputation model to detect malfunctioning or malicious agents in multi agent systems in general and in sensor networks specifically is widely used [20], [17], [4]. In the proposed DCOP_MST model, the importance of coverage (or the requirement) for each point in the given area is represented *by the total sum of agents' credibility which is required in order to cover it*. The model's goal function will be to minimize the difference between the credibility of the agents which are covering the area and the coverage requirements.

The agents in the proposed adaptive framework perform a distributed constraint optimization algorithm in order to select their position in the given area. Due to the dynamic nature of the problem and the large number of possible assignments, a complete algorithm would not be practical for solving DCOP_MSTs. In contrast, most existing incomplete (local) search algorithms for DCOPs can be adjusted to solve a DCOP_MST and some of them are also compatible to cope with the dynamic nature of such problems.

The basic local distributed algorithm proposed in order to solve DCOP_MST is a distributed self adjusting algorithm based on the Maximum Gain Message (MGM) algorithm [10], [14] which is a distributed (message passing) local search algorithm. The quick convergence of the MGM algorithm is an essential property in an environment with intensive changes. Furthermore, MGM is monotonic and thus, is expected to avoid redundant expensive movements of sensors. However, the drawback of the MGM algorithm, being monotonic and incomplete, is the convergence to a local minima which in this application might cause the failure to recognize targets in the area. Our investigation of existing exploration methods to escape local minima reveals their inability to cope with the special requirements of the DCOP_MST model. We propose two exploration methods which allow the mobile agents to explore the area for new targets while maintaining a high level of coverage on the targets which were previously detected.

Our empirical study investigates the performance of MGM_MST with respect to the mobility and sensing limitations of the agents. Existing algorithms which perform exploration were implemented and compared with the proposed exploration methods. Both of the proposed methods outperform the state of the art algorithms by a large factor.

2. Model Description

For lack of space we present only the components of the standard DCOP model. For a full and formal definition the reader is referred to [13].

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.

In order to present the DCOP_MST model, a number of concepts (which are not used in the standard DCOP model) need to be defined. 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 variables 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

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.

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.

3. Solving algorithm for DCOP_MST

After defining a model for representing mobile sensing agents team problems, the next step is to propose algorithms that the agents in the DCOP_MST model will use in order to select their position.

The choice of local search for solving problems of the DCOP_MST model is supported by the common standard considerations for selecting local over complete search which are time limitations and the limit on the size of problems for which complete algorithms are practical. In addition, the special properties of a mobile sensing agent team problem also encourage the choice of a local search algorithm:

1) Exploring the entire search space as required for complete search, would mean that agents take each and every possible position. This does not seem practical for sensors with limited mobility on a large area.
2) The expected dynamic changes limit the expected time agents will have to perform a complete algorithm. In such a dynamic environment, the algorithm is expected to maintain reasonable coverage while adjusting to the changes in the problem [11].

The simplicity of the framework of local search algorithms makes it compatible with a dynamic environment. Many local search algorithms (as MGM and DSA [14], [21]) evaluate only the current state in each iteration, while in complete algorithms, agents consider information which was inferred in previous steps of the algorithm (like Nogoods for example [18]). This information might not be valid after the problem changes.

Among the existing local search algorithms, MGM was selected for its fast convergence and its simplicity. In addition, MGM is monotonic, thus, it would avoid costly redundant movements by agents. We note that the same adjustments we present for MGM are also applicable for many other local DCOP algorithms with a synchronous greedy structure.

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 algorithm is a simpler version of the *DBA* algorithm [19], [21]. 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 [3]. 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 2 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 $Curr\_DIFF$. 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 $Curr\_DIFF$. 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.

Figure 1 presents the code of the MGM_MST algorithm. The main loop of the algorithm remains almost unchanged from standard MGM [10], [14] (the standard algorithm was left out for lack of space). The agents send

MGM_MST
1. $value \leftarrow$ SelectedValue()
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)

Figure 1. MGM_MST.

their assignments (current positions) to the agents which are currently their neighbors. Notice that according to the assumptions in Section 2 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.

3.1. Runtime example

Figures 2 to 4 present an example of a DCOP_MST solved by the MGM_MST algorithm. The team includes five mobile sensors. The dashed lines circling each of the sensors present their sensing range. The mobility range for each agent is considered to be two times the sensing range (this range was left out of the figures in order to simplify the presentation). The agents are required to cover a number of target areas which are depicted by complete circles each containing a number. The number represents the environmental requirement function ER. In the initial state of this example depicted on the left hand side (LHS) of Figure 2 there are three target areas with $ER = 3$ and one with $ER = 10$. The initial credibility assigned to each agent is 5 (it is depicted as a framed digit above each dashed circle). In the initial state presented on the LHS of Figure 2 all targets are covered as required. The ordered set of events OE includes two events. The right hand side (RHS) of Figure 2 presents the state after the first event, possibly a conflict in the report, that triggered a decrease in the credibility of sensors 1 and 2. As a result, the difference between the requirements on the target and the sum of the credibility of agents 1 and 2 is 4. Agent 5 is currently covering a target with ER value of 3. Thus, it moves to a position where it covers the target which is considered more important. The resulting state is presented on the LHS of Figure 3. Agent 4 can improve its local state by moving to a position in which it covers the target it covered before and the target which agent 5 left uncovered (resulting in the state presented on the RHS of Figure 3). Notice, that agent 4 moves although it is already covering an area with $ER = 3$ since even though it is covering the point with the largest current Cur_DIFF value in its range, the recursive function requires it to keep considering the different positions which cover this point and possibly additional points with a similar or less Cur_DIFF value. The LHS of Figure 4 presents the state after an environmental change (the second event in OE). A new target area was added with $ER = 3$. Agent 1 changes its position since its contribution to the coverage of the target it is currently covering is less than its contribution when covering the new target. The final state is presented on the RHS of Figure 4.

4. Theoretical properties and bounds

The first bound that needs to be established is that the local method performed by each agent in each iteration is

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 2 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.

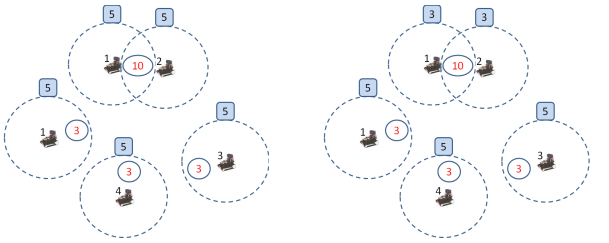

Figure 2. LHS initial state. RHS credibility change for some of the sensors.

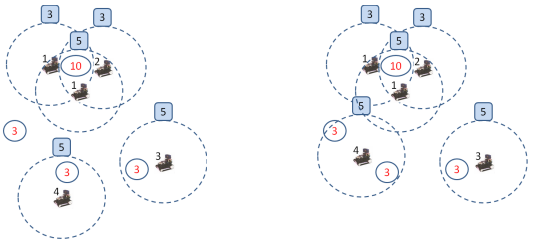

Figure 3. LHS first step of adjustment. RHS second step of adjustment.

efficient. Otherwise, it is not realistic to assume that agents can complete the computation of the optimal alternative position in a single iteration of the algorithm. [3]. The optimality of the local method is essential to insure that the maximal possible gains of agents are indeed exchanged by agents. Otherwise, the algorithm is not guaranteed to converge to a local optimum.

Lemma 1: Assuming the maximal number of possible positions in the MR of an agent is m, the number of calls to method **select_pos** this agent will make at the most in a single iteration of the algorithm is $m+1$ (linear).
proof: When method **select_pos** is called for the first time, the set of possible positions includes m members at the most. Since in each call to the method the values of points which are covered from all possible positions (the $intersection\_area$) are not included in the function (lines 25,26 of Figure 1) the points which will be entered into the next generated $target\_set$ cannot be in sensing range from all possible positions. Since only positions which are in sensing range from all the points in $target\_set$ are entered into the next generated set of possible positions (line 24), in every recursive call, after the first call, the set of possible positions is smaller. Thus, the function can be called $m+1$ times at the most. Notice that the set is never empty according to the second termination condition (lines 20,21). □

Lemma 2: Assuming the maximal number of possible positions in the SR of an agent from any target point is s, the number of calls to method **select_pos** this agent will make in a single iteration of the algorithm is $s+1$ (linear).
proof: In each call of method **select_pos** a target set is generated. Only positions within sensing range from all the points in the target set are considered when the function will be called again. Therefore, after the first call, the set of possible positions generated cannot be larger than the number of positions in SR from the points in the target set. The rest of the proof is similar to the proof of Lemma 1. □

3. In our proof we assume that there are no plateaus (continuous areas with the same ER value) and that the number of points of the same (highest) value can be found efficiently. If plateaus do exist, the proof is still valid only there is a need to use geometric computation in order to evaluate areas instead of points.

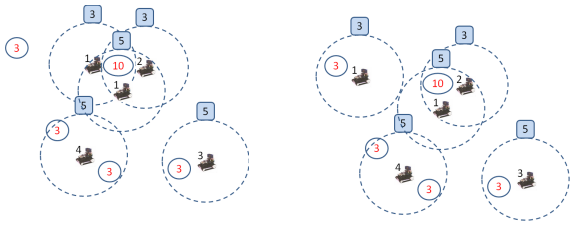

Figure 4. LHS environmental change. RHS final adjustment.

The conclusion from Lemma 1 is that in the worst case, the runtime complexity of a single iteration is: $m^2 * \|target\_set\|$ (m is defined as in Lemma 1) , since in each recursive call, each possible position is checked whether it enables coverage of the $target\_set$. Furthermore, the conclusion from Lemma 2 is that the runtime complexity of a single iteration is $(m+s^2)*\|target\_set\|$ at the most ((m and s are defined as in Lemmas 1 and 2). Thus, the runtime of a single iteration is the minimal among the two [4].

Next, the (local) optimality of the method for selecting an agent's position is established (it is optimal if only the actions of a single agent are considered). As mentioned above, the local optimal selection of position by the agents is essential to ensure the maximal gain property in the MGM algorithm:

Lemma 3: In each recursive call of the **select_pos** method, the set of possible positions includes the optimal position with respect to minimizing the largest value of the Cur_DIFF function, within the sensing range of all the positions that are in the mobility range of the agent.
proof: Assume that the selection of possible positions in the i'th call to the recursive method **select_pos** by agent A_j is the first which does not include the optimal position. We differentiate between two cases:

1) There exists at least one possible position in the possible positions set of the $i-1$ call which is within the sensing range of all the points in the $target\_set$ generated in the $i-1$ iteration.
2) No possible position in the possible positions set of the $i-1$ call is within the sensing range of all points in the $target\_set$ generated in the $i-1$ iteration.

The consequence of the first case is that there exists an optimal position pos' which was included in the possible positions of the $i-1$ call and is not selected to be included in the new set of possible positions. This means that pos' is not within SR_j of all points in the target set (line 24 of Figure 1) . However, the goal is to minimize the Cur_Diff function and the $target\_set$ includes the points with the largest $Curr\_Diff$ values not within sensing range from all possible positions found in the $i-1$ iteration. Thus, the fact that there exists a position which enables coverage of all points in $target\_set$ contradicts the optimality of pos'.

For the second case, any selection of position will give the same largest difference. Thus, any selection is locally optimal and the choice of selecting the position which covers the largest number of points in the $target\_set$ is

4. In contrast to the assumptions made, in case the initial possible position set or target set are too large and the method cannot be completed in reasonable time, the method can be stopped and one of the positions in $target\_set$ can be selected. However, in this case local optimality is not guaranteed.

a heuristic which hopefully would help in most cases to achieve the global goal. □

Since the **select_pos** method returns either a position which was left last in the possible positions set or one position from a set of positions from which the agent does not have any coverage differences, this selection is optimal with respect to the position selection of a single agent. However, we note that (as expected of a local search algorithm) MGM_MST is not guaranteed to reach a global optimal state.

5. Exploration methods

Classic local search combines exploitation methods in order to converge to local optima and exploration methods in order to escape them [22]. The proposed 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 [10], [21], [14]. However, some of the methods which are most effective in standard DCOP are not expected to be effective for DCOP_MST.

For standard DCOPs, a K-opt [14] algorithm gives an upper bound on the distance from the optimal solution. This guarantee is achieved by agents considering all of the problem's constraints (by groups of size K) in every iteration of the algorithm. In DCOP_MST, if a target is not in the range of any agent it will not be considered. Therefore, a K-opt algorithm is expected to allow agents to converge to a deployment which results in better coverage of the targets in range but it cannot offer the same guarantees as in standard DCOPs when there are targets beyond the agents' ranges.

Another method which is most effective for standard DCOPs is the anytime framework proposed in [23]. In this framework, agents are storing the best solution which was explored in memory and this solution is reported when the algorithm is terminated. In DCOP_MST, agents change their physical position and are expected to maintain coverage of targets which were detected. Changing to the best solution can require agents to travel a long distance and at the same time leave targets uncovered. In addition, the method is effective only for static problems since there are no guarantees on the quality of the solution when the problem changes. Thus, holding the best position found so far in memory while exploring for new targets is not expected to be effective for DCOP_MST.

In order to explore the area for new targets while maintaining coverage of targets which were previously detected we propose 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). 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 $Curr\_Diff$ 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 the proposed 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.

6. Experimental evaluation

The proposed DCOP_MST model was evaluated using a simulator representing a mobile sensors team problem. The problem simulated is of an area in which the possible positions are a 200 over 200 grid. Each of the points in the area has an ER value between 0 and 100. The ER function initially included 10 random points with maximal requirement (of 100). The credibility of an agent varies between zero (for an agent with no credibility) and 100 for an agent with maximal credibility. The credibility variable was initially set to 30. The reputation model used in our experiments was inspired by SPORAS [20]. As in SPORAS, all agents are initiated with similar credibility (or "reputation value" [20]) [5] and the effect of the events on the credibility of agents is with respect to their current level of credibility . The set E included three types of events:

1) An environmental event which increases a point in the area to a maximum ER value. This event can represent an intelligence report that some enemy activity is about to happen in a specific point.
2) The credibility of two neighboring agents decreases by 25% (to 75% of what they had before the event). This event can represent a conflict in the reports of the two neighboring agents.
3) The credibility of a single agent decreases by 50%. This event represents an agent which its reports indicate that it is suffering from some technical problem.

Figure 5 demonstrates the effect of the technology, more specifically, the sensing and mobility ranges on the quality of the algorithm. Both sides of the figure show the results of the MGM_MST algorithm after 15 random events. The results depicted are an average over 50 runs of the algorithm. On the LHS of Figure 5 the sensing range is fixed and limited and the mobility range varies.

5. In contrast to SPORAS the initial credibility is not zero since in MSTs we are not concerned with agents using different pseudonyms

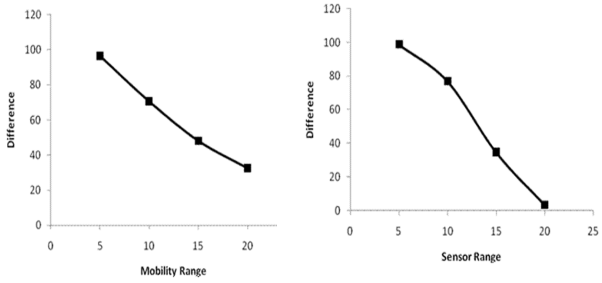

Figure 5. Current Difference for varying ranges.

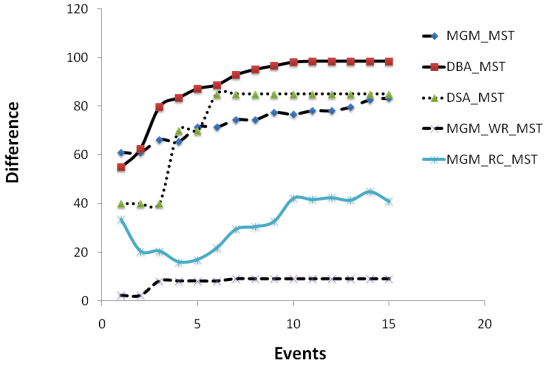

Figure 6. Current Difference for exploration methods.

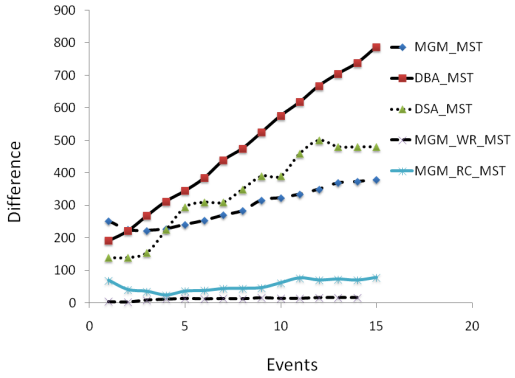

Figure 7. Total amount of difference for exploration methods.

On the RHS of Figure 5 the mobility range is fixed and the sensing range varies. It is clear from these two figures that in order for the MGM_MST algorithm to perform well, at least one of the parameters, either the MR or the SR should be high. Otherwise, the algorithm cannot handle events beyond the agents' ranges and the current difference value remains high. In other words, in order to benefit from the nice properties of the MGM algorithm, e.g. quick convergence and monotonicity, the agents must be equipped with technology that enables either a large sensing range or a large mobility range. When the technology is limited exploration methods are required in order to overcome this drawback.

In order to cope with the limitations of the monotonic algorithm, four different exploration methods were implemented and compared with MGM_MST. The first two methods are versions of the DSA and DBA algorithms [21]. In the first, DSA_MST, agents do not collect

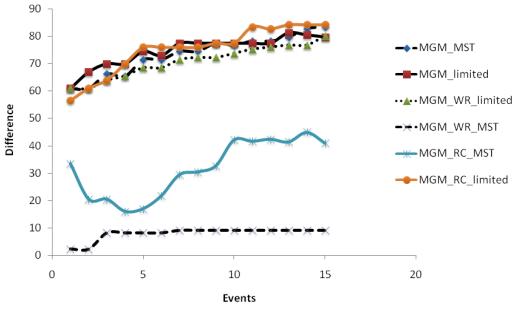

Figure 8. Comparison with the standard model (current difference).

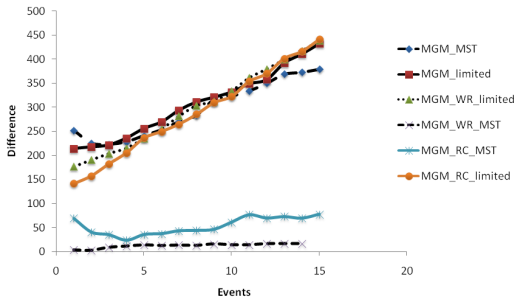

Figure 9. Comparison with the standard model (total amount).

the best reductions from their neighbors. Instead, an agent changes its position to a best alternative that offers a positive reduction according to some probability variable p. In our experiments $p = 0.6$. In the second, DBA_MST, agents which detect that they are in a quasi local minima (i.e., their LR is negative and so is the LR of their neighbors) change the ER function by reducing the value of all the points in their sensing range by one. The other two exploration methods were the proposed methods described in Section 5.

In this set of experiments, the ranges for all agents were $SR = 15$ and $MR = 10$. The results in Figure 6 present the current difference function for all four exploration methods and the MGM_MST algorithm. In this set of experiments, after each of the 15 events, the algorithms ran for 50 iterations and the results presented are the current difference at the end of these 50 iterations. Each point represents an average over 20 runs of the algorithm. These results demonstrate the success of the proposed exploration methods over the classic exploration methods. In order to verify that the success of the proposed methods is not only in the extreme highest difference in the area, a second metric is presented. In Figure 7 the results present the total sum of the current difference function over all the points in the area for the five versions of the algorithm. In this graph, the large improvement of the proposed exploration methods is even more apparent.

In the last set of experiments the importance of the dynamic domains and dynamic sets of neighbors (constraints network) in the proposed model was evaluated. Figures 8 and 9 compare the MGM_MST algorithm and the two exploration methods which were found successful in the experiments of the proposed model (presented in

Figures 6 and 7) with the same algorithm and exploration methods, only using the standard model with a fixed constraints network and fixed domains. The results are very conclusive. When the standard model, with the fixed domains and fixed constraints network is used (in our figures we refer to them as the 'limited' algorithms), the exploration methods are not effective. In fact, both the standard MGM and the algorithms with the exploration methods produce the same results. It is clear that the dynamic elements in the proposed model enable efficient exploration.

7. Conclusions

A new model based on DCOP for solving a dynamic problem was proposed. The proposed model, DCOP_MST, 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. DCOP_MST enables representation of dynamic coverage requirements and the dynamic variability in the quality of agents' reports which can be caused by technology limitations and malicious actions.

A local search algorithm based on MGM was designed to solve problems represented in the proposed model. Agents in the MGM_MST algorithm make an efficient selection of the locally optimal position with respect to the goal function. As an incomplete search algorithm, MGM_MST does not guarantee to converge to a globally optimal solution. However, it converges fast to a local optima and its monotonic property insures that agents avoid redundant movements. The same adjustments which were applied to MGM can be used to adjust other local search algorithms with a similar synchronous structure, in order to solve DCOP_MSTs. Two existing algorithms which perform exploration were implemented and were found to be not effective for the presented model in our experimental study. Two new exploration methods which follow a periodic structure, enable the detection of new targets while maintaining acceptable coverage on the targets which were previously detected.

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] D. A. Burke, K. N. Brown, M. Dogru, and B. Lowe. Supply chain coordination through distributed constraint optimization. In *Proceedings of the 9th International Workshop on Distributed Constraint Reasoning (DCR09)*, Providence, RI, USA, September 2007.

[3] Z. Collin, R. Dechter, and S. Katz. Self-stabilizing distributed constraint satisfaction. *Chicago Journal of Theoretical Computer Science*, 5, 1999.

[4] R. Du, M. Xu, and H. Zhang. An extended hierarchical trusted model for wireless sensor networks. *Wuhan University Journal of Natural Sciences*, 11:1489–1492, 2006.

[5] L. Frye, C. Liang, S. Du, and M.W. Bigrigg. Topology maintenance of wireless sensor networks in node failure-prone environments. In *Proc. IEEE International Conference on Networking, Sensing and Control*, pages 886–891, April 2006.

[6] A. Gershman, A. Grubshtein, L. Rokach, A. Meisels, and R. Zivan. Scheduling meetings by agents. In *10th international workshop on Distributed Constraint Reasoning (DCR08), AAMAS 2008*, Estoril, Portugal, May 2008.

[7] A. Gershman, A. Meisels, and R. Zivan. Asynchronous forward-bounding for distributed constraints optimization. In *Proc. ECAI-06*, pages 103–107, August 2006.

[8] A. Howard, M. J. Matarić, and G. S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots Special Issue on Intelligent Embedded Systems*, 13(2):113–126, 2002.

[9] R. N. Lass, E. A. Sultanik, and W. C. Regli. Dynamic distributed constraint reasoning. In *AAAI*, pages 1466–1469, Chicago, IL, USA, 2008.

[10] 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.

[11] R. Mailer. Comparing two approaches to dynamic, distributed constraint satisfaction. In *AAMAS*, pages 1049–1056, Utrect, Netherlands, 2005.

[12] J. Modi and M. Veloso. Multiagent meeting scheduling with rescheduling. In *Proc. of the Fifth Workshop on Distributed Constraint Reasoning (DCR), CP 2004*, Toronto, 2004.

[13] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: asynchronous distributed constraints optimization with quality guarantees. *Artificial Intelligence*, 161:1-2:149–180, January 2005.

[14] J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *IJCAI*, Hyderabad, India, January 2007.

[15] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 266–271, 2005.

[16] S. Poduri and G. S. Sukhatme. Constrained coverage for mobile sensor networks. In *Proceeding of the IEEE International Conference on Robotics and Automation*, pages 165–171, 2004.

[17] S. D. Ramchurn, D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19:2004, 2004.

[18] M. C. Silaghi and M. Yokoo. Nogood based asynchronous distributed optimization (adopt ng). In *AAMAS*, pages 1389–1396, 2006.

[19] M. Yokoo. Algorithms for distributed constraint satisfaction problems: A review. *Autonomous Agents & Multi-Agent Sys.*, 3:198–212, 2000.

[20] G. Zacharia, R. Moukas, and P. Maes. Collaborative reputation mechanisms in electronic marketplaces. In *HICSS*, 1999.

[21] 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.

[22] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

[23] R. Zivan. Anytime local search for distributed constraint optimization. In *AAAI*, pages 393–398, Chicago, IL, USA, 2008.