Explorative Max-sum for Teams of Mobile Sensing Agents

Harel Yedidsion, Roie Zivan
Dept. of Industrial Engineering and Management, Ben Gurion University
Beer-Sheva, Israel
yedidsio@bgu.ac.il, zivanr@bgu.ac.il

Allesandro Farinelli
Dept. of Computer Science,University of Verona
Verona, Italy
alessandro.farinelli@univr.it

ABSTRACT

Multi-agent applications that include teams of mobile sensing agents are challenging since they are inherently dynamic and a single movement of a mobile sensor can change the problem that the whole team is facing. While agents select their positions with respect to the information available to them in their local environment, by moving to a different location they can reveal new information, e.g., targets, which they were not aware of before. Thus, exploration is required for such information to be revealed. A variation of the DCOP model (DCOP_MST) was previously adjusted to represent such problems along with local search algorithms that were enhanced with exploration methods.

In this paper we design an explorative version of Max-sum for solving DCOP_MST, which is based on an iterative process where, at each iteration, agents generate and solve a specific problem instance. We demonstrate that this basic algorithm (Max-sum_MST) converges faster than other standard local search algorithms that were adjusted to solve DCOP_MSTs, however, its exploitive nature makes it inferior to explorative local search algorithms.

Thus, we designed exploration methods that when combined with basic Max-sum_MST, significantly outperform the existing explorative local search algorithms. Moreover, the best performing method we propose also eliminates the exponential time complexity of Max-sum by bounding the number of agents involved in each constraint.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: [Multi-agent systems]

General Terms

Algorithms, Experimentation

Keywords

DCOP, Incomplete Algorithms, GDL

1. INTRODUCTION

Some of the most challenging applications of multi-agent systems include teams of mobile sensing agents that are required to acquire information in a given area. Examples include networks of sensors that track targets [16] and rescue teams operating in disaster areas [4]. A crucial, common feature of these applications is that agents select physical locations to move to, and that this selection affects their future interactions, e.g., if a mobile sensor decides to sense a given area, it will then coordinate its actions with nearby sensors, which it might not have considered (or even met) before.

These types of scenarios have been previously modeled using the Distributed Constraint Optimization Problem (DCOP) framework by representing mobile sensors as agents that need to select locations and their tasks/targets as constraints [9]. However, if all possible future moves of dynamic agents are considered, the problem becomes dense and hence, finding an optimal solution in reasonable time becomes infeasible. Thus, previous work deals with the inherent dynamism of such scenarios by suggesting an iterative process. In each iteration a DCOP instance is built representing the current situation (e.g., sensor positions) and in which only limited movements of the agents are considered. Agents run a distributed algorithm (that might involve several communication cycles) to decide what would be the best next joint move. After they execute the selected joint move, they build a new DCOP instance considering their new positions [9]. This approach generates inherent locality for agents, i.e., in each iteration an agent only considers alternative positions it can move to (in this iteration) and tasks it can fulfill (targets it can cover) when located at these positions.

The result of such a design may be that some tasks are not considered by enough agents, e.g., targets are not within sensing range from locations that sensors are considering to move to (i.e., not within the local environment of enough sensors with respect to the targets' coverage requirements) and thus, even if the sensors exploit their local information optimally, the team's deployment may be of low quality. In such cases some agents are required to explore for tasks beyond their local environment. In such a scenario, a move to what appears to be a low quality position may turn out to be most effective. Hence, agents may not be aware of the actual reward for some of their possible moves, (as in [10]).

Recently, Zivan et al. [16] proposed a model and corresponding local search algorithms for representing and solving such scenarios, particularly focusing on teams of mobile sensing agents that need to select a deployment for the sensors in order to cover a partially unknown environment - *DCOP_MST*. DCOP_MST is an extension of the DCOP model that allows agents to adjust their location in order to adapt to dynamically changing environments. The local distributed search algorithms that were proposed for solving DCOP_MST, were adjustments of standard local search techniques (such as Maximum Gain Message (MGM) [5] and Distributed Stochastic Algorithm (DSA) [14]) to the model, enhanced by specifically designed exploration methods [16].

Nevertheless, prior to this work, only local search algorithms were proposed for solving DCOP_MST. This is in spite of the intensive study on incomplete inference algorithms, (e.g., Max-

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

sum), that were applied to many realistic applications including mobile sensor networks [9] and teams of rescue agents [7]. In contrast to standard local search algorithms, agents in Max-sum do not propagate assignments but rather calculate utilities (or costs) for each variable, considering all possible value assignments of their neighboring agents' variables. The general structure of the algorithm is exploitive, i.e., the agents attempt to compute the best costs/utilities for possible value assignments according to their own problem data and recent information they received via messages from their neighbors. However, exploration requires that agents perform non exploitive actions.

The need for exploration can be reduced by extending the local environments of the agents and allowing them to consider further away tasks/targets. However, this would increase the number of agents that can be assigned to each task. Since the computation performed by Max-sum is exponential in the number of agents involved in a constraint, constraints that involve many agents (*k-ary*) represent a computational bottleneck. While a number of techniques were proposed to reduce such complexity [9, 4], in general settings, the time required to perform such computation is exponential in the constraint arity. Thus, an increase in the number of agents that can be assigned to tasks would prevent the use of Max-sum for solving such problems.

Taking all the above into consideration, in this work we focus on adjusting the Max-sum algorithm to the DCOP_MST model by designing efficient exploration methods that allow agents to select sub optimal positions and seek for additional targets that are currently beyond their sensing ranges. In more detail, in this paper we contribute to the state of the art first by applying the periodic incremented largest reduction (PILR) approach, which was found successful for local search algorithms, to Max-sum. In this approach, periodically (i.e., every k iterations where k is a predefined number), agents are allowed to select sub optimal positions. As a result, agents can select locations from which they can be effective for covering targets beyond their current local environment defined by their mobility and sensing ranges.

Next, we propose a novel exploration method, specifically designed for Max-sum, based on meta-reasoning: agents select for each target a subset of the sensors that can be effective for covering it. The size of the subset is equal to the maximal number of sensors required for covering the target. This target is ignored in the process for selecting the locations of other sensors. As a result, such sensors that were not selected for coverage of targets are free to explore for new targets.

The proposed function meta reasoning method (FMR) breaks the relation between the size of the local environment of agents and the arity of the constraints, i.e., the arity of the constraint is not defined by the number of sensors that can be within sensing range of a target t after the next assignment selection (i.e., the "neighbors" of t) but rather by the required number of sensors for covering t. Thus, even if we enlarge the local environment of agents and the number of neighbors of target t grows, the number of neighbors for t in the reconstructed factor-graph is bounded from above by the number of sensors required for covering it. Our empirical study reveals that a greedy heuristic for selecting the subset of the neighboring sensors for coverage improves the performance of the method further.

We empirically compare the proposed exploration methods and the adjusted iterative version of standard Max-sum to existing local search methods for DCOP_MST. Our results demonstrate that standard Max-sum is superior to standard local search algorithms (in terms of iterations to reach convergence and solution quality) but it is outperformed by local search algorithms that include exploration methods. However, when Max-sum is combined with any of the exploration methods described, it outperforms the explorative local search algorithms, and the combination of Max-sum with function meta reasoning dominates all other approaches. Moreover, we demonstrate that an increase in the size of the local environments of agents does not affect the runtime required for completing an iteration for agents performing the FMR method while the runtime required for agents to complete an iteration in all other methods based on Max-sum grows exponentially.

The rest of the paper is organized as follows: Section 2 discusses previous work, while Section 3 describes the DCOP_MST model and the existing leading solution algorithms. Section 4 presents the adjustment of Max-sum for solving DCOP_MSTs (i.e., Max-sum_MST) and Section 5 describes the exploration methods we propose. Finally, Section 6 describes our experimental study and Section 7 concludes the paper.

2. RELATED WORK

DCOP is a general model for distributed problem solving that has been widely used to coordinate the activities of cooperative agents [5, 14, 11, 8]. The DCOP literature offers a rich wealth of solution techniques, ranging from complete approaches [6], which are guaranteed to find the optimal solution, to heuristic methods [14, 15, 8, 12] that do not provide optimality guarantees but can provide high quality solutions for systems of significant magnitude (in terms of number of agents and constraint density). Since DCOPs are NP-hard, heuristics are typically preferred for practical applications where a solution must be returned within a few seconds.

A number of papers proposed the DCOP model for representing and solving coordination problems related to sensor networks [1] and mobile sensor networks [9]. Specifically, Stranders et al. in [9] focus on a mobile sensor placement problem, where a network of sensors must cooperatively measure a scalar field (e.g., temperature) to minimize the measurement uncertainty. The method proposed in that work includes the construction of a DCOP instance for every selection of a limited path for the sensors to follow and gather the information required. The Max-sum algorithm is used for solving the DCOP and coordinating sensors' movements. Similar to that work, in order to adjust Max-sum to DCOP_MST we also build a DCOP instance based on the current sensors' positions and use the Max-sum algorithm to solve each instance. However, while in [9] agents share a model of the environment that provides them with an estimate of the reward associated to each joint move, in our scenario agents need to explore their surroundings as information beyond their local environment might significantly change their reward (e.g., the discovery of a new task). Therefore, we introduce explicit mechanisms for exploration for the Max-sum algorithm.

The concepts of exploration and exploitation have been investigated, in the DCOP context, by Taylor et al. in [10]. They consider a specific setting where mobile sensors have knowledge about the distribution of rewards in the environment, but they do not know the exact rewards, and they can not perform an exhaustive exploration of all the actions. Consequently, authors propose a series of approaches to address the trade-off between exploring new (possibly sub-optimal) actions versus exploiting the best actions experienced so far. A crucial difference with respect to our work is that the topology of the network (and hence the constraint graph of the DCOP) does not change when agents move, while the dynamism of the DCOP (and particularly of the constraint graph) is a key component for the DCOP_MST model and for the solution techniques that we propose here.

Finally, regarding the Max-sum algorithm, there is a significant body of work that focuses on different aspects of the algorithm such as, convergence guarantees [8, 17], evaluation for realistic

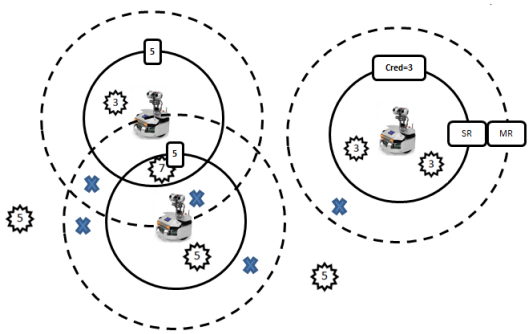

Figure 1: An example of the MST coordination problem.

applications [7] and computational complexity [4, 3]. Our work contributes to this ongoing effort by extending the applicability of Max-sum to teams of mobile sensing agents, by providing exploration mechanisms.

3. BACKGROUND

In this section we provide the necessary background on 1) coordination for mobile sensor teams. 2) the DCOP_MST model and the associated solution techniques.

3.1 Mobile Sensor Teams

In a mobile sensor team, agents $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ are physically situated in the environment, which is modelled as a metric space with a distance function d. We assume that the locations that can be occupied by agents are a set of discrete points that form a subset of the total environment, and the current position of agent A_i is denoted by cur_pos_i. Moreover, time is discretised into an undeterminate series of time-steps, and the maximum distance that A_i can travel in a single time step is defined by its *mobility range* MR_i. Figure 1 illustrates the relevant aspects of the model for an exemplar situation, agents are depicted by small robots, and the possible locations are shown by "X"s. The dashed, outer circles centered on the agents represent their mobility range and all "X"s within the circle are locations that the agent can move to in a single time step.

As for perception, agents have limited, heterogeneous sensing ranges (SR_i denotes the sensing range of agent A_i), and each agent can only provide information on targets within its sensing range. Moreover, agents may also differ in the quality of their sensing abilities, a property termed their *credibility*. The credibility of agent A_i is denoted by the positive real number $Cred_i$, with higher values indicating better sensing abilities[1]. Figure 1 reports the sensing ranges of each agent (the inner circle centered at the agent) and the credibility, shown by the number on each sensing range circle.

The individual credibilities of agents sensing the same target are combined using a *joint credibility function* $F : 2^A \rightarrow \mathbb{R}$, where 2^A denotes the power set of A. We require F to be monotonic so that additional sensing agents can only improve the joint credibility. Formally, for two sets $S, S' \subseteq A$ with $S \subseteq S'$, we require that $F(S) \leq F(S')$. For simplicity, in the remainder of this paper we will use the sum function to aggregate agents' credibilities:

$$F_{sum}(S) = \sum_{A_i \in S} Cred_i$$

[1]We assume that $Cred_i$ is exogenously provided (for instance, calculated by a reputation model) and accurately represents the agent's sensing ability; dealing with inaccurate scores is of interest but beyond the scope of the work.

Targets are represented implicitly by the *environmental requirement* function ER, which maps each point in the environment to a non-negative real number representing the joint credibility required for that point to be adequately sensed. In this representation, targets are the points p with $ER(p) > 0$. A major aspect of the mobile sensing team problem is to explore the environment sufficiently to be aware of the presence of targets. In Figure 1 there are a number of targets (stars) and their numbers represent their ER values.

Agents within SR of a target are said to *cover* the target and the *remaining coverage requirement* of the target, denoted Cur_REQ, is the environmental requirement diminished by the joint credibility of the agents currently covering the target, with a minimum value of 0. Denoting the set of agents within sensing range of a point p by $SR_{(p)} = \{A_i \in A[d(p, cur\_pos_i) \leq SR_i]\}$, this is formalized as $Cur\_REQ(p) = \max\{0, ER(p) \ominus F(SR_{(p)})\}$, where $\ominus : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is a binary operator that decreases the environmental requirement by the joint credibility.

The global goal of the agents is to position themselves so to minimize the values of Cur_REQ for all targets. In some cases it may be possible to reduce the values of Cur_REQ to zero for all targets indicating perfect coverage. However, in other cases this may not be possible (e.g., because of insufficient numbers or quality of agents). In these cases, we aim at minimizing the sum of remaining coverage requirements for all targets. Such a minimization problem is NP-hard [13].

3.2 Distributed Constraint Optimization

A distributed constraint optimization problem (DCOP) is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ where $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ is a finite set of agents , $\mathcal{X} = \{X_1, X_2, \ldots, X_m\}$ is a finite set of variables, $\mathcal{D} = \{D_1, D_2, \ldots, D_m\}$ is the set of finite domains for the variables, and $\mathcal{C}$ is a finite set of constraints. Each variable X_i is controlled (or owned) by an agent who chooses a value to assign it from the finite set of values D_i; each agent may control multiple variables. Each constraint $C \in \mathcal{C}$ is a function $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}_+ \cup \{0\}$ that maps assignments of a subset of the variables (called the scope of the constraint) to a non-negative *cost*. The cost of a *complete assignment* of values to all variables is computed by summing the costs of all constraints. The goal of a DCOP is to find a complete assignment with minimum cost (or with maximal utility in the case of a maximization problem).

Control in DCOPs is distributed, with agents only able to assign values to variables that they control. Furthermore, agents have knowledge only of the constraints involving variables that they control. Coordination is achieved through message passing. A standard assumption is that an agent can exchange messages only with a subset of the other agents, called *neighbors*. Two agents are neighbors if there is at least one constraint connecting the variables that the agents control. While transmission of messages may be delayed, it is assumed that messages are received in the order that they were sent.

3.3 The DCOP_MST Model

The DCOP_MST model is a dynamic DCOP formulation that models the mobile sensor team coordination problem [16]. DCOP_MST takes the dynamic nature of such an application into account by reacting to changes as they occur, instead of requiring an explicit model of the dynamics.

Specifically, each agent A_i controls one variable that represents its position, and the domain of such a variable contains all locations within MR_i of cur_pos_i; consequently, as the agent moves from one location to another, the content of its variable's domain changes. A change in the content of a domain of some variable can

induce a change in the constraints that include it. This is because only agents that can take a value within sensing range of a target are included in the constraint that calculates the coverage for this target. Hence, the constraint C_p for a target p, only involves those agents A_i, whose variable's domain includes a location within SR_i of p. Therefore, as the domains change, the constraints change as well. As a consequence the set of neighbors for each agent changes over time as the agents move. In more detail, in DCOP_MST two agents are neighbors if their sensing areas overlap after they both move as much as possible in a single time step towards each other. This encodes the fact that such two agents might directly influence each other (e.g., by observing the same target in the next time step).

The *local environment* of agent A_i is the joint area within SR_i from all positions within MR_i from cur_pos_i, i.e., it includes all targets that the agent can cover after a single move.

3.4 Adjusting local search to DCOP_MST

For lack of space we do not include a detailed description of the implementation of DSA and MGM in DCOP_MST.[2] We note that standard local search algorithms such as DSA and MGM require that agents will be able to compute the best alternative assignment for their variable. In DCOP_MST, the best alternative is a position that allows the coverage of the targets with the highest current coverage requirement in range.

As mentioned before, a crucial element for solving a DCOP_MST problem is to provide the agents with a mechanism for exploring potentially sub-optimal solutions that might result in increased reward for the team (e.g., because the agents discover a target that was previously not included in their local environment).

A successful exploration mechanism must be able to balance the exploration with the exploitation of the local knowledge that can produce solutions with low costs. In other words, agents should explore the area for new targets while maintaining coverage of targets that were previously detected. To this end, a number of powerful methods were proposed in [16], and the most successful one is a periodic strategy named: Periodic Incremented Largest Reduction (PILR).

The PILR approach allows agents, in some iterations, to select sub-optimal assignments, such as a joint move that results in an increase of the Cur_REQ function up to a constant bound c. Such a strategy is applied periodically, i.e., every k_1 iterations, agents can perform sub-optimal moves for k_2 iterations. The parameter c controls the increase in cost that agents are willing to accept when performing the sub-optimal moves and, by tuning this parameter, we allow agents to search for new targets while preventing them from abandoning the ones they are aware of. The most successful technique according to [16] was the combination of PILR with DSA (DSA_PILR).

3.5 Standard Max-sum

The Max-Sum algorithm [2] is a GDL algorithm that operates on a *factor graph*: a bipartite graph where the nodes represent variables and constraint functions [2, 17].[3]

In more detail, in a factor graph, each variable-node is connected to all function-nodes that represent constraints with which it is involved.[4] Similarly, a function-node is connected to all variable-nodes that represent variables included in the scope of the constraint it represents.

Both variable and function nodes are active computational entities that can send, read and update messages. The agents are then responsible for the computation associated to the nodes and, in more detail, each variable-node is assigned to the agent controlling this variable, and each function-node is assigned to one of the agents controlling the variables that form the scope of the associated constraint function.

The content of messages sent by function-nodes differs from the content of messages sent by variable-nodes. A message sent from a variable-node to a function-node includes, for each of its possible value assignments, the sum of costs/utilities for this value reported by all other function neighbors. A message sent from a function-node to a variable-node in each step includes, for each possible value assignment of the variable, the best (minimal in a minimization problem, maximal in a maximization problem) cost/utility that can be achieved for any combination of assignments to the variables involved in the function, apart from the destination variable. At the end of the run, each variable node selects the value assignment that has the best sum of costs/utilities, considering the most up to date messages received from all neighboring function-nodes.

4. APPLYING MAX-SUM TO DCOP_MST

Applying Max-sum to DCOP_MST is challenging since Max-sum does not propagate assignments but rather utilities (or costs). While assignment selections are not a part of the Max-sum algorithm, assignment selections determine the local environments in DCOP_MST and directly affect the structure of the constraint network (and consequentially, the factor graph).

We overcome this obstacle by following the scheme proposed in [9], which is an iterative process in which in each iteration the agents construct a factor graph based on their current location, run the Max-sum algorithm for a number of message cycles and move according to the solution provided by the algorithm. Agents select the locations from which they derive the highest utility, i.e., from which they are most effective (ties are broken randomly). The next factor graph is generated considering the new locations of the agents.

The number of message cycles that are performed before an assignment (position) selection, must be selected with care. On one hand, we would like to allow the information regarding the coverage capabilities of sensors to propagate to other sensors. On the other hand, these message cycles of Max-sum result in a single movement for the sensors, thus, we want to avoid unnecessary delays. In our experiments we found that the any-time performance of Max-sum (i.e. the best result it finds throughout the search [15]) converges very fast and thus, a small number of message cycles (5 in our experimental set-up) was enough to get the best performance.

The time complexity for the message update operations performed by the function-nodes in Max-sum in each message cycle is known to be exponential in the size of the function scope (i.e., the arity of the constraint/function). In the DCOP_MST model, a function-node represents a target, and the arity of the function is the number of sensors that can sense the target after a single move. Therefore, for scenarios where sensors have large sensing and mobility ranges, the arity of constraints can be large (in the worst case it could be equal to the number of agents) ,hence, the time requirement for message computation of function nodes can be a severe bottleneck when using Max-sum for solving such DCOP_MSTs..

A number of papers proposed techniques to reduce the complexity of the message update calculation for function-nodes in Max-sum [9, 4]. Here, we implemented all of the methods proposed in

[2]A detailed description can be found at [16].

[3]For lack of space we describe Max-sum briefly. The reader is referred to the following papers for a detailed description of the algorithm [2, 17].

[4]We preserve in this description the terminology used in [2], and call constraint-representing nodes in the factor graph "function-nodes".

these papers including: i) preprocessing the variables' domains to detect and eliminate values that are dominated by others (proposed in [9]). In our experiments, this preprocessing method pruned between 25% to 30% of the value assignments; ii) mapping the domains of all variables to two values that are required for performing the function computation (proposed in [4][5]). This is possible because the only information required to compute the value of a function is whether a sensor covers the target or not; iii) using branch and bound search to calculate the utility for a value assignment of a neighboring sensor instead of a naive search through all possible assignment combinations (proposed in [9]).

It is important to note that while the methods mentioned above significantly reduce the complexity of the calculation performed for production of messages in the algorithm the calculation performed by the function-nodes is still exponential. Thus, the effective number of neighboring sensors that a target can have is still limited (and small).

5. EXPLORATION METHODS FOR MAX-SUM_MST

While Max-sum_MST is superior to standard local search (see experiments presented in Section 6), it is outperformed by algorithms that include explicitly designed exploration methods.

We use the example presented in Figure 2 to demonstrate the need for exploration in Max-sum_MST and the use of the proposed exploration methods. It includes three agents (sensors) (A_1, A_2 and A_3) and two targets (T_1 and T_2). The factor graph on the right represents agents as variables and targets as functions (V_1, V_2, V_3, F_1 and F_2 respectively). The credibility of all sensors is 5 and targets have importance 10. Two sensors (A_1 and A_2) are currently fully covering T_1. A_3 can cover T_1 if it would make a single move in its direction, therefore V_3 is connected to F_1 in the factor graph. In contrast, none of the sensors can be in sensing range from T_2 after a single move, therefore, F_2 is disconnected in the factor graph.

In contrast to local search algorithms, e.g., DSA, in which when a target is fully covered, other agents would not consider covering it too, in Max-sum the highest utility is propagated by the targets for each possible location in its mobility range, considering all possible locations of the other sensors in range. Thus, in the case of A_3, F_1 would send a message to V_3 assigning the maximal possible contribution A_3 would have in covering T_1. In doing so, the function would consider the case in which A_1 and A_2 would move to further locations and allow A_3 to be effective in T_1's coverage. This behavior would prevent A_3 from exploring even though T_1 is fully covered. This phenomenon can be avoided using the exploration methods we propose.

Thus, in this section we propose exploration methods for Max-sum_MST that aim to allow agents to explore for new targets while maintaining coverage on targets they have previously detected.

5.1 Max-sum_PILR

In our first attempt to introduce exploration into Max-sum_MST, we aimed to duplicate the success of the periodic exploration methods that were combined with local search algorithms. Similar to the MGM_PILR and DSA_PILR algorithms, Max-sum_PILR encourages exploration by allowing agents periodically to select a sub-optimal assignment. More formally, Max-sum_PILR is defined by three parameters k_1, k_2 and c. After k_1 iterations in which it performs its standard operations, the algorithm performs k_2 iterations in which each agent selects a random position among the possible positions from which the utility (the effectiveness of the agent) was within c from the utility it would have derived if it would have selected the best possible position (the position that would have been selected in a standard iteration). Formally, for every variable-node x representing a sensing agent, the agent calculates for every value v in its domain a utility $u(v) = \sum_f u_f(v)$ where f is a neighboring function-node and $u_f(v)$ is the utility associated with value v that was included in the last message received from f. Assume that v' is the value (position) in the domain of x with maximal calculated utility, i.e., $\forall v \in D_x, u(v') > u(v)$. Thus, the agent selects a position for x randomly among all values $v \in D_x$ for which $u(v) \geq u(v') - c$. The agents perform periodically k_1 standard iterations and k_2 random selection iterations as described above.

[5]This method is also known as: "Fast Max-sum"

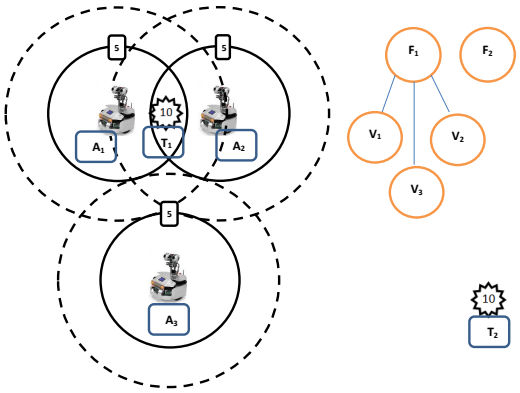

Figure 2: An example of the exploration process.

If we return to the example presented in Figure 2, in Max-sum_PILR, the agents would periodically consider sub-optimal positions from which they would select their position randomly. Hence, it is now possible for A_3 to move to a location that will allow it to include T_2 in its local environment.

5.2 Max-sum_FMR

The second approach for exploration we propose is based on function meta reasoning, and therefore it is termed Max-sum_FMR. This approach takes advantage of a property that is quite common in DCOP_MST, that targets have more neighbors than required for covering them. Consider an iteration i in which the factor graph FG_i was generated based on the locations of sensors selected in iteration $i-1$. Denote by $n(t)_i$ the number of neighboring sensors that target t has in FG_i and by $r(t)$ the maximal number of sensors that are required to cover t. When $r(t) < n(t)_i$, t can select $r(t)$ neighbors for covering it and allow the other $n(t)_i - r(t)$ neighbors to perform exploration. We implement this by generating a new factor graph $\widehat{FG}_i$ in which each target t has at most $r(t)$ neighbors. This can be done distributively by having each target t remove the edges between it and $n(t)_i - r(t)$ of its neighbors.

Figure 3 displays a section of a factor graph in which function-node F_1 has 6 neighboring variable-nodes ($V_1, , V_6$). In this example function-node F_1 represents target T_1, which has a coverage requirement of 100. The 6 variable-nodes represent 6 neighboring mobile sensors ($n(T_1) = 6$), each with credibility 40. The function used for combining agents' coverage capabilities is the standard additive function F_{sum}. Therefore, three sensors are required at most for covering this target ($r(t) = 3$). Thus, in $\widehat{FG}_i$, F_1 will have three neighboring variable-nodes (sensors) for which it will compute their best position using standard Max-sum_MST, while its other three neighboring variable-nodes in FG_i are disconnected from it and therefore the sensors they represent are encouraged to explore.

It is important to notice that when using this method, the com-

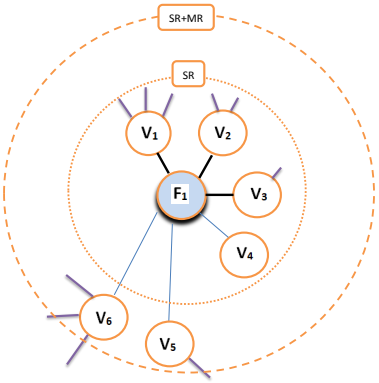

Figure 3: An example of Max-sum_FMR.

plexity for producing each of the messages to be sent by the function-node to its neighbors is no longer exponential in $n(t)_i - 1$ as in standard Max-sum, rather it is exponential in $r(t) - 1$. Thus, the complexity of the computation of function-nodes is no longer dependent on the sensing and mobility ranges of the sensors. In other words, this method eliminates the main drawback of Max-sum compared to local search algorithms.[6]

The selection of the covering neighbors by the method can affect its success. If a sensor that is selected by a target moves to a position such that the target is beyond its sensing range, this target will remain uncovered. This can happen if multiple targets select a sensor that cannot cover all of them from a single location. Thus, the agent will need to select a position from which it covers only part of the targets that selected it. Such a selection may result in a poor outcome.

We propose the following greedy heuristic for selecting the $r(t)$ neighbors by a function-node t for which $n(t)_i > r(t)$:

1. Each of the $n(t)_i$ sensor neighbors sends to t its degree in FG_i (i.e., the number of function-node neighbors it has in FG_i).

2. t divides its $n(t)_i$ neighbors into two subsets: $\hat{n}(t)_i$ and $\bar{n}(t)_i$. $\hat{n}(t)_i$ includes all neighbors that are currently located within sensing range from t and $\bar{n}(t)_i$ includes the rest of the neighbors.

3. While $(n(t)_i > r(t))$

   (a) If $(\bar{n}(t)_i \neq \emptyset)$ remove the neighbor in $\bar{n}(t)_i$ that has the highest degree from $n(t)$.

   (b) Else, remove the neighbor in $\hat{n}(t)_i$ that has the lowest degree from $n(t)$.

In the example presented in Figure 3, according to the heuristic proposed, the first variable node to be removed from the set of neighbors of F_1 is V_6, which among the variable-nodes out of SR is the one with the highest degree. The second to be removed is V_5, which is also out of SR. V_4 will also be removed from the set although it is within SR since there are no more neighboring variable-nodes out of sensing range and it is the one with the lowest degree among the rest.

[6]This advantage is not limited to DCOP_MST. In fact, it can be applied to any task allocation application where the property $r(f) < n(f)$ is common (where $r(f)$ is the required number of neighbors for the task represented by f and $n(f)$ is the actual number of neighbors), e.g., allocation of rescue teams to tasks in a disaster area, grid computing etc.

The heuristic above attempts to increase the probability that the neighbors that are selected for coverage are indeed able to cover the targets that selected them. It does so by preferring sensors that are currently within sensing range. Obviously, if it was possible for every target t to select $r(t)$ sensors that are currently located within sensing range, then Max-sum_FMR would have produced the optimal solution. Thus, the heuristic prefers sensors within sensing range, and among them, the sensors that are effective for other targets as well (i.e., with a higher degree). However, if there are no more neighbors within sensing range, the selected neighboring sensor will need to move closer in order to be effective, thus, among the neighbors outside of sensing range it selects the least constrained sensors. Our empirical results demonstrate its advantage over a random selection in settings where sensors may be selected by more than one target (i.e., when agents have larger local environments).

In the problem depicted in Figure 2, Max-sum_FMR will have T_1 select one of its neighbors and remove the edge connecting them in the factor graph. Notice, that according to the heuristic we propose, A_3 is the neighbor it will disconnect since it is the only one that is not currently covering it (i.e., the only one in $\bar{n}(T1)$). Thus, the chances that A_3 will move towards T_2 are increased.

6. EXPERIMENTAL EVALUATION

In order to compare Max-sum_MST and its versions that include exploration with the DCOP_MST algorithms proposed in [16], we used a simulator representing a mobile sensing agent's team problem. The problem simulated is of an area in which the possible positions are an m over m grid. Each of the points in the area has an ER value between 0 and 100. The mobility and sensing ranges are given in terms of distance on the grid and are varied in our experiments to demonstrate their effect on the success of the algorithms. The credibility of an agent can vary between zero (for an agent with no credibility) and 100 (for an agent with maximal credibility). The method for calculating the joint coverage of agents within the sensing range of a target is a standard sum of the agents' credibility and the $\ominus$ operator is a standard minus, i.e.: $Cur\_REQ(p) = \max\{0, ER(p) - \sum_{A_i \in SR_{(p)}} Cred_i\}$
The credibility variable in the experiments for all agents was set to 30. These values were chosen so targets with maximal importance (100) will require the cooperation of multiple agents.

Max-sum algorithms ran 5 rounds of messages in every iteration. This number was found to produce best results.[7] All results depicted in this section are an average over 50 runs of the algorithm solving 50 different random problems.

Figure 4 presents a comparison between Max-sum_MST, the standard local search DCOP algorithms that were adjusted to DCOP _MST in [16] and the PILR explorative algorithms. This experiment included 50 agents (sensors) and 20 targets that were deployed randomly in a 100 over 100 grid. Each target had importance 100. Thus, the sum of all uncovered targets is at most 2000. The sensing range (SR) and the mobility range (MR) were set to 5. The results demonstrate the advantage of Max-sum_MST over standard local search algorithms. Not only does it converge in a smaller number of iterations[8], but its final result is also better than standard DCOP algorithms. However, it is clearly inferior in com-

[7]A smaller number of message rounds produced inferior results, while for a larger number we experienced loopy propagation behavior as described in [17], which again, produced lower quality solutions.

[8]Notice that each Max-sum iteration takes multiple message rounds, i.e., more time to compute than in standard local search.

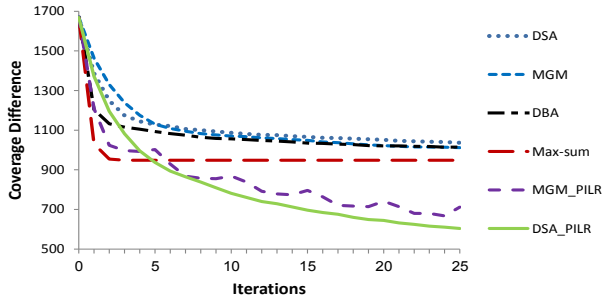

Figure 4: Sum of coverage differences over all targets as a function of the number of iterations. SR=5, MR=5

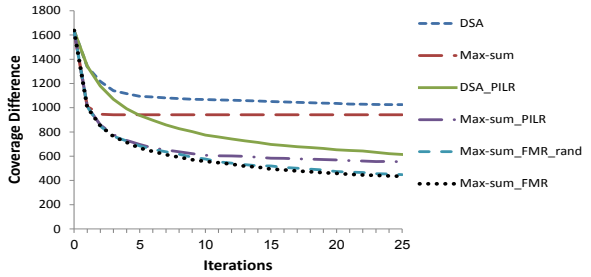

Figure 5: Sum of coverage differences over all targets, as a function of the number of iterations. SR=5, MR=5.

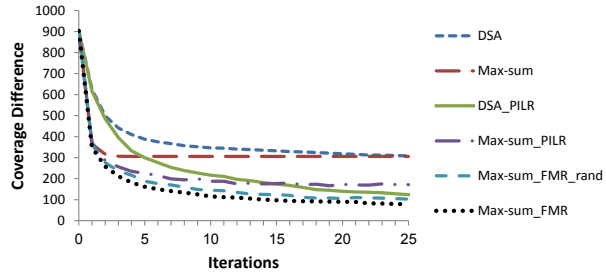

Figure 6: Sum of coverage differences for all targets, as a function of the number of iterations. SR=10, MR=5.

parison with local search algorithms that are combined with exploration methods.

In the second set of experiments we evaluated the performance of the versions of Max-sum_MST that include exploration (see Section 5), Max-sum_PILR and Max-sum_FMR. These exploration algorithms are compared with standard Max-sum_MST, DSA and DSA_PILR. The settings in this experiment were the same as in the experiment described above. The parameters used in Max-sum_PILR were: $k_1 = 4$, $k_2 = 1$ and $c = 20$, i.e., every fifth iteration, a random position is selected from the set of positions whose current utility is within 20 from the value assignment with the maximal current utility. The value of 20 for the parameter c was chosen both for Max-sum_PILR and for DSA_PILR. For Max-sum_FMR the sensors credibility values and the importance of targets indicated that no more than 4 sensors are needed to cover a target, i.e., $r(t) = 4$. In order to evaluate the impact of the heuristic for selecting neighbors to be removed from the factor graph, presented in Section 5, we also depict the results of Max-sum_FMR-rand in which a random selection of neighbors was performed.

The experiments presented in Figure 5 included sensors with $SR = 5$ and $MR = 5$. It is apparent that Max-sum based algorithms reach solutions with low coverage difference (high quality) within a smaller number of iterations than local search algorithms. Moreover, the resulting solutions that Max-sum's exploration methods produce, in terms of coverage, are superior to existing local search algorithms even when they are combined with exploration methods. Specifically, Max-sum_PILR produces better results than DSA_PILR and Max-sum_FMR produced the best results.[9] Figures 6 and 7 present results for experiments in which the local environments of agents were larger. In Figure 6, the ranges were $SR = 5$ and $MR = 10$ and in Figure 7, $SR = 10$ and $MR = 10$. It is apparent that when agents have larger local environments the need for exploration is reduced. However, Max-sum_FMR dominates on all types of problems. Interestingly, the impact of the selection heuristic for Max-sum_FMR is most significant when the environments are the largest. Our investigation revealed that when environments are small there is less of a conflict between targets, i.e., most sensors do not have multiple targets within their environment and therefore, they are not selected by more than one target. For example, the average number of neighboring targets for a sensor in experiments with $SR = 5$ and $MR = 5$ was 1.3 and for experiments with $SR = 10$ and $MR = 5$ it was 2.1. Thus, both selection methods will commit most sensors only to a single target and therefore, a significant advantage of the proposed heuristic over the random selection is not expected. On the other hand in experiments with $SR = 10$ and $MR = 10$ the average number of neighboring targets for a sensor was 3. Thus, scenarios in which a sensor is selected by multiple targets but there is no location that it can select within sensing range from all these targets, are more common.

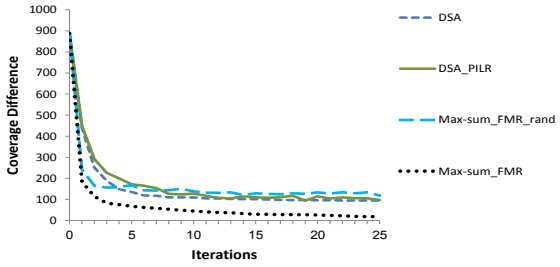

Figure 7: Sum of coverage differences for all targets, as a function of the number of iterations. SR=10, MR=10.

Figure 8 presents the results of the algorithms when solving much smaller problems for which we were able to produce the optimal solutions using exhaustive search. These problems included a 20 over 20 grid, 8 sensing agents with $SR = MR = 3$ and 4 targets. It is evident that the explorative versions of Max-sum find higher quality solutions than the explorative local search algorithm when solving small problems as well. They find solutions with a ratio as low as approximately 1.4 (or 140%) of the optimal solution.[10]

Figure 9 presents a runtime comparison between the algorithms. The runtime was calculated by adding for each synchronous message round of the algorithm the maximal time it took an agent to complete its actions. The results indicate the runtime required to complete 50 random experiments for varying sensing ranges. It was shown in [16] that given large enough sensing and mobility ranges, even the simplest myopic algorithms can quickly achieve optimal coverage. We assume that if Max-sum could have run with larger ranges it would produce high quality results as well. How-

[9]The differences between Max-sum_FMR and DSA_PILR were found to be significant. However, the advantage of the use of the FMR edges selection heuristic was only found to be significant when solving high density problems.

[10]Here we do not present the results for the random selection version of Max-sum_FMR since in small problems with a small number of agents such selection heuristic has no effect.

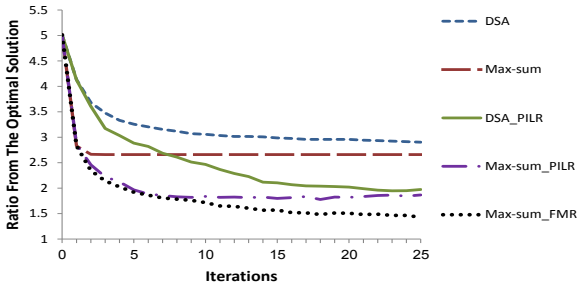

Figure 8: Ratio from the optimal solution, as a function of the number of iterations for different algorithms. Small problems.

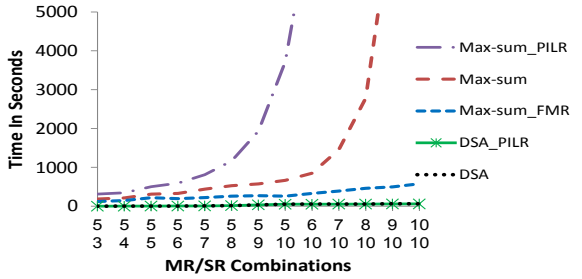

Figure 9: The time required to complete 50 experiments as a function of different sensing and mobility ranges.

ever, our analysis illustrates that Max-sum cannot benefit from such improved technology since larger sensing and mobility ranges imply more neighboring agents per target and therefore exponentially greater computation efforts. The MR in the experiments was varied between 5 and 10 and the SR was varied between 3 and 10. Figure 9 demonstrates the exponential growth of runtime as a function of the sensing and mobility ranges (notice that the sum of both ranges on the x axis is increasing linearly).

It is apparent that while the runtime of Max-sum_MST and Max-sum_PILR grows exponentially, the increase in time when using DSA and DSA_PILR is negligible. As we expected, Max-sum_FMR performs in a near linear runtime while producing the best results in terms of coverage quality.

It is also notable that Max-sum_PILR performs much slower than standard Max-sum_MST. A further investigation revealed that the maximal number of neighboring sensors that targets have in Max-sum_PILR was double than in Max-sum_MST. Apparently, the increased exploration of Max-sum_PILR allows agents to detect targets and cluster around them. This emphasizes the need for Max-sum_FMR, which triggers exploration but is not exponential in the number of neighboring sensors that targets have.

7. CONCLUSION

In this paper we adjusted the Max-sum algorithm to the DCOP_MST model by designing efficient exploration methods that allow agents to select sub optimal positions and seek for additional targets that are currently beyond their sensing ranges. Specifically, we proposed two classes of exploration methods that can be combined with Max-sum_MST. The first (Max-sum_PILR) implements the periodic reduction of requirements approach that was found successful for local search algorithms. The second (Max-sum_FMR) required function-nodes (targets) to perform meta-reasoning and manipulate some of the sensors to perform exploration. Our empirical evaluation shows that this class of methods not only produces the best results in terms of coverage but also eliminates the relation between the size of the local environment (commonly defined by sensors' technical limitations, i.e., their mobility and sensing ranges, which affect the arity of the constraints) and the complexity of the calculation performed by the function nodes. As a result, we show that, in contrast to the other versions of Max-sum, Max-sum_FMR does not exhibit an exponential increase in time when the number of sensors that can be effective for each target grows.

8. REFERENCES

[1] A. Farinelli, A. Rogers, and N. Jennings. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *AAMAS*, 2013.

[2] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS*, 2008.

[3] Y. Kim and V. R. Lesser. Improved max-sum algorithm for dcop with n-ary constraints. In *AAMAS*, 2013.

[4] K. S. Macarthur, R. Stranders, S. D. Ramchurn, and N. R. Jennings. A distributed anytime algorithm for dynamic task allocation in multi-agent systems. In *AAAI*, 2011.

[5] R. T. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for dcop: A graphical-game-based approach. In *PDCS*, 2004.

[6] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: asynchronous distributed constraints optimizationwith quality guarantees. *Artificial Intelligence*, 2005.

[7] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. Decentralized coordination in robocup rescue. *Computer Journal*, 2010.

[8] A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 2011.

[9] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *IJCAI*, 2009.

[10] M. E. Taylor, M. Jain, Y. Jin, M. Yokoo, and M. Tambe. When should there be a "me" in "team"?: distributed multi-agent optimization under uncertainty. In *AAMAS*, 2010.

[11] M. Vinyals, J. A. Rodríguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming dcop algorithms via the generalized distributive law. *AAMAS*, 2011.

[12] X. S. W. Yeoh and S. Koenig. Trading off solution quality for faster computation in dcop search algorithms. In *IJCAI*, 2009.

[13] G. Wang, G. Cao, P. Berman, and T. F. Laporta. A bidding protocol for deploying mobile sensors. In *in Proceedings of IEEE ICNP*, 2003.

[14] 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*, 2005.

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

[16] R. Zivan, R. Glinton, and K. Sycara. Distributed constraint optimization for large teams of mobile sensing agents. In *International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 2009.

[17] R. Zivan and H. Peled. Max/min-sum distributed constraint optimization through value propagation on an alternating dag. In *AAMAS*, 2012.