Engineering Applications of Artificial Intelligence 71 (2018) 87–99

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Applying max-sum to teams of mobile sensing agents

Harel Yedidsion [a,*], Roie Zivan [a], Alessandro Farinelli [b]

[a] Industrial Engineering and Management Department, Ben-Gurion University of the Negev, Beer Sheva, Israel
[b] Computer Science Department, University of Verona, Verona, Italy

ARTICLE INFO

Keywords:
Distributed constraint optimization
Incomplete algorithms
GDL
Max-sum
Multi-agent systems
Exploration
Mobile sensor networks

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 entire team is facing. A variation of the Distributed Constraint Optimization model for Mobile Sensor Teams (DCOP_MST) was previously adjusted to represent such problems along with local search algorithms that were enhanced with exploration methods. This paper considers the use of the Max-sum algorithm for solving problems of deploying a mobile sensor team in an unknown environment to track and monitor points of interest (targets), represented by the DCOP_MST model.

The DCOP_MST model allows the representation of different functions for aggregating the joint coverage of targets by multiple sensors. The use of different functions has a dramatic effect on the complexity of the Max-sum algorithm. When using cardinality functions, Max-sum can be performed efficiently regardless of the arity of constraints. When Max-sum is used to solve applications that require other (more complex) aggregation functions, its complexity is exponential in the arity of the constraints and thus, its usefulness is limited.

In this paper we investigate the performance of the Max-sum algorithm on two implementations of the DCOP_MST model. Each implementation considers a different *joint credibility function* for determining the coverage for each target, with respect to the locations and the credibility of agents. In the first, the coverage is calculated according to the number of agents that are located within sensing range from the target. This function can be calculated efficiently. The second takes the angle between the lines of sight of different agents to a target into consideration. The larger the difference in the angle between the lines of sight, the higher the coverage efficiency.

We analyze the challenges in adjusting the Max-sum algorithm in both scenarios and propose enhancements of the algorithm that make it more efficient. We provide empirical evidence of the advantages resulting from these enhancements in comparison to the naive algorithm.

1. Introduction

As development of autonomous robots rapidly expands, alongside sensor, actuation and communication technology, it is likely that soon, teams of mobile sensing agents would be commonly used to perform various collective tasks. Some challenging applications of Mobile Sensor Teams (MSTs) include tracking and monitoring points of interest in an unknown environment (Lesser et al., 2012; Zivan et al., 2015), measuring a scalar field (La and Sheng, 2013), maintaining wireless sensor networks (Hermelin et al., 2017), and creating a communication network (Jain et al., 2009). In other applications, MST's form rescue teams operating in disaster areas (Macarthur et al., 2011; Pujol-Gonzalez et al., 2013). Examples of underwater data collection using autonomous underwater robotic vehicles include monitoring of algal blooms (Smith et al., 2010), seismic activity (Nooner and Chadwick, 2009), measurement of ocean currents (Hollinger et al., 2016) and schools of robotic fish monitoring pollution in waterways (Hu et al., 2011). Moreover, the advancement of the internet-of-things (IoT) technology provides the necessary infrastructure for mobile sensors to become smart agents, which can share information and coordinate their actions (Rust et al., 2016). In such a setting, where a large number of mobile agents need to cooperate, it is important to have efficient protocols for communication, task allocation, deployment and decision-making.

MSTs are inherently decentralized as each agent has exclusive control of its own location and has limited computational and communication resources. As the number of agents increases, these limitations necessitate that computation and communication be distributed over the

* Corresponding author.
E-mail addresses: yedidsio@bgu.ac.il (H. Yedidsion), zivanr@bgu.ac.il (R. Zivan), alessandro.farinelli@univr.it (A. Farinelli).

https://doi.org/10.1016/j.engappai.2018.02.017
Received 11 January 2017; Received in revised form 5 February 2018; Accepted 19 February 2018
Available online 20 March 2018

entire team to avoid a single point of failure, communication bottlenecks or unacceptably long delays.

Modeling distributed multi agent systems is often done using the Distributed Constraint Optimization (DCOP) framework (Maheswaran et al., 2004b; Meisels, 2008; Yeoh et al., 2008; Le et al., 2016) (Section 3.1 offers a formal description of the framework).

Recently, Zivan et al. (2015) 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*. The DCOP_MST model 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) Maheswaran et al., 2004a and Distributed Stochastic Algorithm (DSA) Zhang et al., 2005) to the model, enhanced by specifically-designed exploration methods (Zivan et al., 2015). The need for reasonable response times drives agent to only consider alternative positions in their local environment. This locality in turn, generates the need to enhance the algorithms with exploration methods that enable agents to consider suboptimal positions in order to escape local minima and find targets outside of their local environment.

The Max-sum algorithm has been the subject of intensive study in DCOP problems and has been applied to many realistic applications including mobile sensor networks (Stranders et al., 2009; Vargo et al., 2013), supply chain management (Chli and Winsper, 2015) and teams of rescue agents (Ramchurn et al., 2010). Max-sum is an incomplete inference algorithm, which propagates costs/utilities, unlike incomplete local search algorithms in which agents share their selected assignments with their neighbors (Zivan et al., 2014). While on random synthetic problems, Max-sum is outperformed by local search algorithms, in many realistic scenarios, such as sensor network scenarios, Max-sum was found to have an advantage (Farinelli et al., 2008, 2013; Stranders et al., 2009; Voice et al., 2010). This motivates the efforts to apply Max-sum to DCOP_MST and evaluate its performance in realistic mobile sensor scenarios, which can be modeled by DCOP_MST.

The need for exploration can be reduced by extending the local environments of the agents and allowing them to consider more distant 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 (Stranders et al., 2009; Macarthur et al., 2011), they are not applicable to every implementation.

Thus, in this work we apply the Max-sum algorithm to two implementations of the DCOP_MST model. Each implementation uses a different function for calculating the joint coverage of a target by the agents that are located in sensing range from it. The first (F_{Sum}), simply adds the credibilities of agents in range. Thus, the optimal joint coverage for this target can be calculated efficiently (Tarlow et al., 2010; Pujol-Gonzalez et al., 2013). The second, (F_{PP}), takes into consideration the angle between the line of sights of agents to the target, assuming that sensing a target from the same angle produces the same information and the larger the difference in the angle between their lines of sight, the more unique information each sensor can provide. This assumption is most common when vision sensors (e.g., cameras) are used (Vazquez et al., 2003; Erdem and Sclaroff, 2006). For this scenario, targets computation is exponential in the arity of the constraint (number of neighboring agents) as in the general case.

We contribute to the state of the art first by applying the Max-sum algorithm to a complex scenario in which it encounters symmetry problems and in which standard runtime enhancement techniques fail to work. We then offer novel solutions to both the symmetry problem and to the runtime enhancement.

The application of Max-sum to F_{PP} necessitates solving the symmetry problem generated by the exploitive nature of Max-sum. We solve this problem by suggesting an efficient local version of the Ordered Value Propagation technique (Zivan and Peled, 2012).

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 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 FMR 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). Section 4.5 explains the symmetry problem in Max-sum and our proposed solution. Section 4.6 describes the exploration methods we propose. Finally, Section 5 describes our experimental study and Section 6 concludes the paper.

2. Related work

The problem of coordinating distributed sensor networks has been solved using a wide range of techniques ranging from bio-inspired behaviors (Xiang and Lee, 2008; Leitão et al., 2012; Das et al., 2014) and machine learning techniques (Wang and de Silva, 2008), to economic and game-theoretic mechanisms (Hsieh, 2009). Other modeling approaches, geared towards software agents, utilize agent-based technology (Aiello et al., 2009; Fortino and Galzarano, 2013). A number of papers proposed the DCOP model for representing and solving coordination problems related to sensor networks (Farinelli et al., 2013; Nguyen et al., 2014) and mobile sensor networks (Stranders et al., 2009).

DCOP is a general model for distributed problem solving that has been widely used to coordinate the activities of cooperative agents (Maheswaran et al., 2004a; Zhang et al., 2005; Rogers et al., 2011; Li et al., 2016). The DCOP literature offers a rich wealth of solution techniques, ranging from complete approaches (Modi et al., 2005), which are guaranteed to find the optimal solution, to heuristic methods (Zhang

et al., 2005; Zivan, 2008; Rogers et al., 2011; Yeoh et al., 2009; Wang et al., 2007) 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.

Inference algorithms such as Max-sum (Farinelli et al., 2008) are often used to coordinate distributed sensor networks. Stranders et al. in Stranders et al. (2009) focus on a mobile sensor placement problem, where a network of sensors must cooperatively measure a scalar field 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 by the sensors. 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 Max-sum to solve each instance. However, while in Stranders et al. (2009) 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 (Taylor et al., 2010; Stranders et al., 2012). Specifically, Taylor et al. in Taylor et al. (2010) considers specific settings where mobile sensors have knowledge about the distribution of rewards in the environment, but they do not know the exact rewards, and they cannot perform an exhaustive exploration of all the actions. Consequently, the authors of Taylor et al. (2010) 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. On a similar line of research, Stranders and colleagues in Stranders et al. (2012) propose an approach to learn the utility function while acting in a stochastic settings. In more detail, the proposed approach is a regret minimization method based on the multi-armed bandit framework. A crucial difference of both these approaches with respect to our work is that they do not consider possible changes in the topology of the network (and hence the constraint graph of the DCOP) when agents act. In contrast, the dynamism of the elements of the DCOP (and particularly of the constraint graph) is a key component for the DCOP_MST model and serves a s key motivation for the solution techniques that we propose here. In this perspective, there are several approaches that propose the use of Dynamic DCOPs to represent problems where the underlying structure (i.e., number of variables, constraints, topology etc.) can change over time (Petcu and Faltings, 2005; Zivan et al., 2015; Nguyen et al., 2014). Specifically, Nguyen et al. in Nguyen et al. (2014) propose a Markovian model for Dynamic DCOPs, explicitly addressing the sequential decision making nature of such problem. However they do not consider possible variable addition/removal or constraint changes. In contrast, the authors of Nguyen et al. (2014) focus on changes that affect features of the domain that compose the state of the underlying MDPs (e.g., the positions of the sensors in the target tracking application they consider).

With respect to such line of research, our approach is based on the DCOP_MST model proposed by Zivan et al. (2015) where the constraint of the problem changes based on the of the value assignments in the previous time step. We propose a solution approach which extends the ability of the Max-sum algorithm to efficiently operate in this setting.

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 (Rogers et al., 2011; Zivan and Peled, 2012), evaluation for realistic applications (Ramchurn et al., 2010) and computational complexity (Macarthur et al., 2011; Kim and Lesser, 2013; Tarlow et al., 2010). Our work contributes to this ongoing effort by extending the applicability of Max-sum to teams of mobile sensing agents, by providing exploration and speedup mechanisms.

This paper is an extension of an earlier conference version of this work (Yedidsion et al., 2014), which presented the first attempt to enhance the Max-sum algorithm with exploration methods. The exploration methods proposed were presented and implemented in combination with the standard Max-sum algorithm, which uses exponential computation (in the arity of the constrains) in order to produce messages. However, many of the relevant applications for DCOP_MST can be solved using the efficient version of Max-sum that uses the THOP method in order to avoid exponential computation. In this paper we discuss two implementations of DCOP_MST, one that can be solved with the efficient version of Max-sum and one that cannot. We discuss the differences between the compatibility of the exploration methods we design between the two and report experimental results for both cases.

3. Background

3.1. Distributed constraint optimization

Distributed constraint optimization is a general formulation of multi-agent coordination problems. A distributed constraint optimization problem (DCOP) is a tuple $\langle \mathcal{A}, \mathcal{X}, D, C \rangle$ representing agents, variables, domains, and constraints respectfully. Each variable X_i is controlled 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 $C \in C$ is a function $C : D_{i_1} \times D_{i_2} \times \cdots \times D_{i_k} \rightarrow \mathbb{R}_+ \cup \{0\}$ that maps assignments of a subset of the variables to a non-negative *cost*. An optimal solution of a DCOP is a complete assignment of minimum cost. The distributed coordination is achieved through message passing between neighboring (constrained) agents.

3.2. The DCOP_MST model

The DCOP model makes several assumptions which do not hold in mobile sensor team applications. It assumes that the domains, neighbor sets and constraints and utilities are known a priori and are constant. Mobile sensors on the other hand are dynamic by nature. The movement of the agents constantly changes these parameters. DCOP_MST formally integrates these dynamic elements into the model. Next, we describe the main concepts of the model. A comprehensive definition of the model can be found in Zivan et al. (2015). In a mobile sensor team, agents are physically situated in the environment. Specifically, each agent A_i controls one variable, denoted by cur_pos_i, that represents its position. Moreover, time is discretized into time-steps, and the maximum distance that A_i can travel in a single time step is defined by its *mobility range* MR_i. Therefore, the domain of agent A_i's position 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_t for a target t, only involves those agents A_i, for which t is within *sensing range*, SR_i from a location included in their variable's domain. 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.

Fig. 1 (Taken from Yedidsion et al., 2014) illustrates the relevant aspects of the model, agents are depicted by small robots. The dashed, outer circles centered on the agents represent their mobility range and all "X"s within the circle are possible locations that the agent can move to in a single time step. As for perception, agents have limited, heterogeneous sensing ranges, and each agent can only provide information on targets within its SR. The agents' quality of their sensing abilities is a property termed *credibility*. The credibility of agent A_i is denoted by the positive real number $Cred_i$, with higher values indicating better sensing abilities. Fig. 1 reports the sensing ranges of

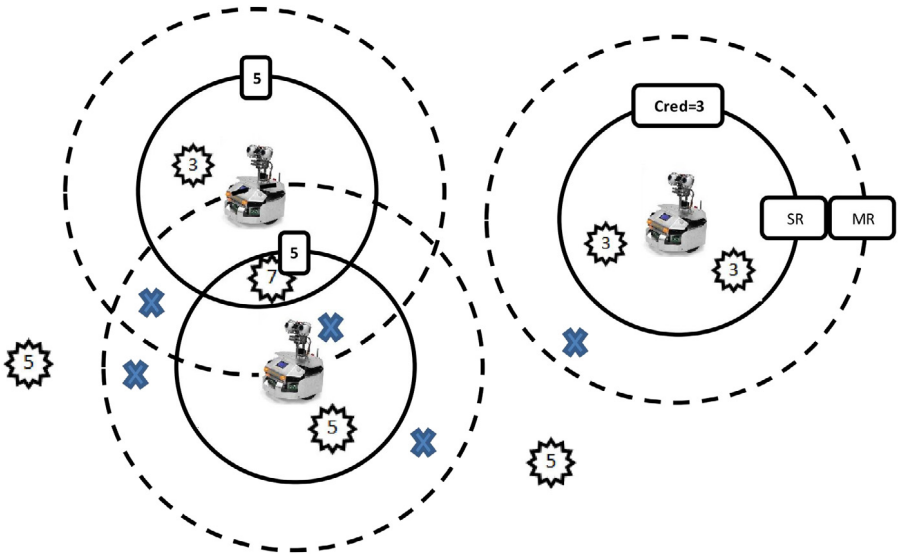


Fig. 1. An example of the MST coordination problem.

each agent (the inner circle) and the credibility, shown by the number on each sensing range circle. 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 $ER(p) > 0$. In Fig. 1 there are a number of targets (stars) and their numbers represent their ER values. Agents within SR of a target t are said to *cover* the target and the *remaining coverage requirement* of the target, denoted $cur\_req(t)$, is the environmental requirement at the location of t 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 $SR_{(p)} = \{A_i \in A[d(p, cur\_pos_i) \leq SR_i]\}$. $cur\_req(p)$ 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, $\min \sum_t cur\_req(t)$. Such a minimization problem is NP-hard (Wang et al., 2003).

3.3. Mobile sensor teams interaction methods

The DCOP_MST model is defined in a generic format, which allows different interpretations of the interaction between agents. Two crucial elements of the model must be precisely defined to apply the model to a specific problem. The first element is *joint credibility function* F which is the function that combines the credibility of neighboring agents and the second is the $\ominus$ operator, which defines how to subtract the joint credibility of the agents currently covering a target from the ER. F is required to be monotonic so that additional sensing agents can only improve the joint credibility. In this paper we examine two implementations of F: the sum function F_{sum} and the proximity penalty function F_{PP}. In both implementations we use the standard subtraction as our $\ominus$ operator. The sum function is defined as follow:

$$F_{sum}(S_t) = \min(cur\_req(t), \sum_{A_i \in S} Cred_i).$$

To define the F_{PP} function we must first describe a number of parameters. The design of this function has been inspired by Abramson et al. (2005), which describes the Prey/Predator problem where predator agents must coordinate their locations around a prey in order to capture it. In an MST application a similar scenario requires mobile sensors to coordinate their locations in order to effectively cover a target. In the F_{sum} implementation on the other hand, the utility that agents derive from covering a target is only affected by the cardinality of the set of agents within sensing range from the target. The PP problem is more realistic when visual sensors are used, e.g., cameras, where the direction from which the target is viewed affects the information that is exposed to the sensor. Thus, two adjacent sensors are expected to produce similar information while sensors that view the target from different angles will produce additional information. The agents must therefore surround the target in uniform spacing to be most effective in covering it. Formally, the following parameters are used by F_{PP}:

Definition 1. A viewing angle between agents A_i and A_j on target t, is the (smallest) angle between the line of sight connecting the location of t and A_i, and the line of sight connecting the locations of t and A_j.

Definition 2. For a target t, MA_t is the minimal viewing angle that does not reduce the credibility of the agents involved in it for covering t.

Denote by S_t the set of agents within sensing range from target t. For each agent $A_i \in S_t$, denote by A_{i_c} its closest clockwise neighbor in S_t, by $A_{i_{cc}}$ its closest counterclockwise neighbor, and by Ang_{i_c} and $Ang_{i_{cc}}$ the smallest viewing angles between A_i and these two agents with respect to t, respectively.

Definition 3. A_i's utility from covering t is calculated using the proximity penalty factor, $PP_{i_t} \in [0, 1]$. If either Ang_{i_c} or $Ang_{i_{cc}}$ are smaller than MA_t, then the factor is less than one. Formally, PP_{i_t} is calculated as follows:

$$PP_{i_t} = [\min(\frac{Ang_{i_c}}{MA_t}, 1) + \min(\frac{Ang_{i_{cc}}}{MA_t}, 1)]/2.$$

Definition 4. A_i's utility from covering target t is:

$$U_{i_t} = Cred_i \cdot PP_{i_t}.$$

The definitions of the parameters above allow us to enclose the formal definition of the F_{PP} function for combining the utilities of all the agents in S_t:

$$F_{PP}(S_t) = \min(cur\_req(t), \sum_{A_i \in S_t} U_{i_t}).$$

We monitor the value of the objective function throughout several iterations of the algorithms. We analyze the speed of convergence and the final outcome.

Fig. 2, demonstrates how four agents should optimally position themselves in order to cover a target efficiently, when all four are needed

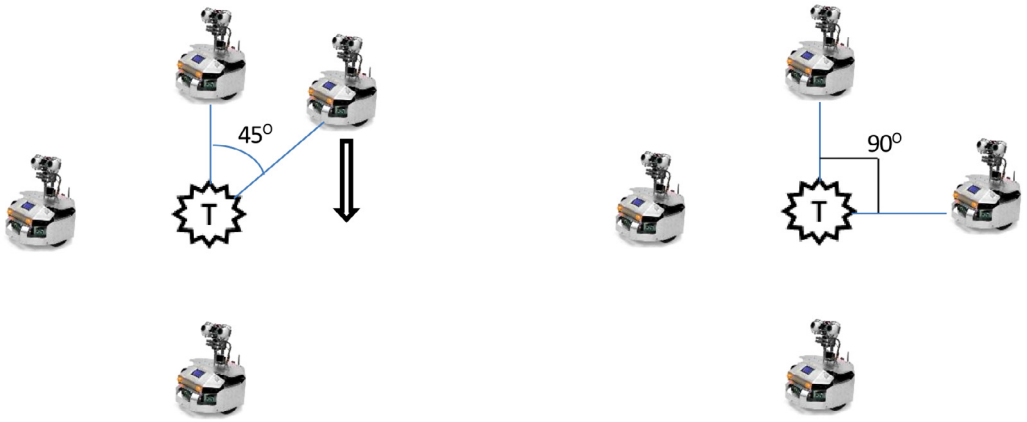


Fig. 2. An example of the DCOP_MST model with the F_{PP} implementation.

to fully cover this single target. In the example depicted in Fig. 2, $N = 4$ and $MA_t = 90$. In the LHS the agents are not optimally located. There is a 45° viewing angle between two of them with respect to the target. This reduces each agent's effectiveness by 25% and the overall coverage of the target by 12.5%. On the RHS the locations of the agents are depicted after one of them moved. Here, the agents are optimally positioned with respect to the coverage of the target.

3.4. Local search for DCOP_MST

Standard local search algorithms were adjusted to DCOP_MST (a detailed description of the implementation of DSA and MGM for DCOP_MST can be found at Zivan et al., 2015). Adjustments include designing a method for detecting the best alternative assignment, considering a different domain and a different set of neighbors in each iteration, and adding a mechanism for exploring potentially sub-optimal solutions. A number of powerful methods were proposed in Zivan et al. (2015), and the most successful one is a periodic strategy named: Periodic Incremented Largest Reduction (PILR). PILR 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. The most successful algorithm proposed in Zivan et al. (2015) was the combination of PILR with DSA (DSA_PILR).

3.5. Standard Max-sum

The Max-Sum algorithm (Farinelli et al., 2008) is a Generalized Distributive Law (GDL) algorithm that operates on a *factor graph*: a bipartite graph where the nodes represent variables and constraint functions (Farinelli et al., 2008; Zivan and Peled, 2012). In a factor graph, each variable-node is connected to all function-nodes that represent constraints with which it is involved.[1] Similarly, a function-node is connected to all variable-nodes that represent variables included in the scope of the constraint it represents. Variable-nodes and function-nodes are considered "agents" in Max-sum, i.e., they can send and receive messages, and perform computation. Their role is actually performed by the DCOP agents. Each agent takes the role of the variable-nodes that represent its own variables and the role of a function-node is performed by one of the agents involved in the constraint it represents. A message sent to or from variable-node x (for simplicity, we use the same notation for a variable and the variable-node representing it) is a vector of size $|D_x|$ including a cost for each value in D_x. In the first iteration all messages include vectors of zeros. A message sent from a variable-node x to a function-node f is formalized as follows:

$$Q^i_{x\to f} = \sum_{f'\in F_x, f'\neq f} R^{i-1}_{f'\to x} - \alpha,$$

where $Q^i_{x\to f}$ is the message variable-node x intends to send to function-node f in iteration i, F_x is the set of function-node neighbors of variable-node x and $R^{i-1}_{f'\to x}$ is the message sent to variable-node x by function-node f' in iteration $i-1$. The values in the messages grow as they keep adding up with every message round. α is a constant that is reduced from all costs included in the message (i.e., for each $d \in D_x$) in order to prevent the costs carried by messages throughout the algorithm run from growing arbitrarily. The value of α may be the lowest value in the message which is reduced from all the costs. Such a choice maintains the differences in costs between assignments. A message $R^i_{f\to x}$ sent from a function-node f to a variable-node x in iteration i includes for each value $d \in D_x$: $max_{PA_{-x}} cost(\langle x, d\rangle, PA_{-x})$, where PA_{-x} is a possible combination of value assignments to variables involved in f not including x. The term $cost(\langle x, d\rangle, PA_{-x})$ represents the cost of a partial assignment $a = \{\langle x, d\rangle, PA_{-x}\}$, which is:

$$f(a) + \sum_{x'\in X_f, x'\neq x, \langle x', d'\rangle \in a} Q^{i-1}_{x'\to f}.d',$$

where $f(a)$ is the original cost in the constraint represented by f for the partial assignment a, X_f is the set of variable-node neighbors of f, and $Q^{i-1}_{x'\to f}.d'$ is the cost that was received in the message sent from variable-node x' in iteration $i-1$, for the value d' that is assigned to x' in a. x selects its value assignment $\hat{d} \in D_x$ following iteration k as follows:

$$\hat{d} = \underset{d\in D_x}{\operatorname{argmin}} \sum_{f\in F_x} R^k_{f\to x}.d.$$

4. Applying Max-sum to DCOP_MST

The success of Max-sum in the coordination of mobile agents (Stranders et al., 2009; Voice et al., 2010; Farinelli et al., 2013) has encouraged our efforts to apply it to DCOP_MST. However, the standard Max-sum algorithm requires several modifications in order to efficiently solve DCOP_MST, including:

1. Adding assignment selection
2. Defining the number of communication rounds
3. Defining the function's computation method
4. Applying runtime reduction methods
5. Handling the symmetry problem
6. Adding exploration methods
7. Defining the tie breaking method.

[1] We preserve in this description the terminology used in Farinelli et al. (2008), and call constraint-representing nodes in the factor graph "function-nodes".

We start by discussing the modifications which must be applied before Max-sum can be used for solving DCOP_MST, and then discuss the design dilemmas, which often require a balance between conflicting objectives in order to achieve the desired result.

4.1. Adding assignment selection

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 Stranders et al. (2009), 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. The next factor graph is generated considering the new locations of the agents.

4.2. Deciding the number of message rounds

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 Zivan, 2008) converges very fast and thus, a small number of message cycles (5 in our experimental set-up) was enough to get the best performance.

4.3. Function computation methods

In standard Max-sum, the message sent from a function f to a variable x, $R_f \rightarrow v$ includes for each value in D_v, a cost/utility specified for a selected combination of assignments to variables by the constraint represented by f. Since constraints are not explicitly defined in DCOP_MST (but rather they are derived from the ER function), the cost/utility values that are calculated in order to generate messages need to be formalized for DCOP_MST. There seem to be two options for the calculation of the function for a specific position in the agent's domain. The first, *max-contribution*, is to specify the maximal contribution for the agent if located in this position, i.e., considering all other agents are as far as possible from the target that f is representing. The second, *max-coverage*, is the maximal coverage for this target, considering the other agents that can be within sensing range, i.e., considering all agents getting as close as possible to the target. Both of these options have limitations which we discuss here and will relate to in the following sections.

In cases where there is a redundancy of agents covering a target, the utility passed in messages by the *max-coverage* function suggests that the target would be covered whether the agent is within SR of the target or out of SR from it. Thus the message from the function to the value contains the maximum utility for both assignments. in this situation the agents are all indifferent towards covering the target, and this may create a situation where neither of them would actually move to a position that allows coverage of the target. In *max-contribution* on the other hand, the function sends to each variable node the utility derived when other agents are out of covering range, i.e., the utility the agent derives when covering the target by itself. Using this form of function calculation, the messages would clearly distinguish between the utility in SR and out of SR. In this case the agents would not be indifferent, and all would move towards the target. This solves the problem of non coverage at the cost of redundancy. This function calculation creates clusters of agents around targets, which are unnecessary and may increase the algorithm's runtime, which is exponential in the number of neighboring agents per target. This also enhances the need for adding exploration techniques to help redundant agents escape this cluster and explore for other less covered targets.

4.4. Runtime reduction methods

The time complexity for the message update operations performed by 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 more details, if the number of variables involved in a constraint F is K, the complexity for generating a message to a variable node x_i (one of these K involved variables) is $O(|D_i|^K)$ (D_i is the domain of variable x_i). In DCOP_MST 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, i.e., sensors that their distance from the target is less than $SR + MR$. 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 techniques were proposed for reducing the complexity of the message update calculation by function-nodes in Max-sum (Stranders et al., 2009; Macarthur et al., 2011; Tarlow et al., 2010). However, not all methods are applicable to all forms of the F function implementations, and while they significantly reduce the complexity of the calculation performed by the function-nodes, it is still exponential. Recently a new method has been proposed by Tarlow et al. (2010) for reducing the complexity of generating messages by function-nodes for specific problems. This method, *Tractable High Order Potentials* (THOP), was adjusted to DCOPs, and implemented by Pujol-Gonzalez et al. (2013). The proposed method reduces the runtime of the Max-sum algorithm to $O(Klog(K))$, but, is only applicable to problems with binary domains where the joint credibility function is a cardinality function. These are special cases of problems where the decision made by agents is on partitioning the variables into groups (e.g., a binary decision of whether the variable/agent is active or not active in covering a target) and the decision is only dependent on the number of variables that are in each group. The implementation using F_{Sum} of DCOP_MST is such a problem. For each target, the assignments of all its neighboring sensors can be categorized to two sets: the set of assignments that are within SR from the target and the set of assignments that are not. The level of coverage is only dependent on the number of agents that are in the first set; more specifically, on the sum of their credibilities. In the implementation using F_{PP} on the other hand, the angles between the lines of sight from each location must be considered, therefore the value assignments of each agent cannot be divided into two sets as required above. Thus, the THOP method cannot be used in this case, nor can the Fast Max-sum method proposed in Macarthur et al. (2011).

4.5. Handling symmetry

The messages from function-nodes to variable-nodes in Max-sum provide for each value in the variable's domain, the maximal possible utility considering all combinations of assignments to all the other variable-nodes involved in the constraint (sensors that can cover the target following a single move). The need to optimize for each of the neighboring agents generates a symmetry that has a dramatic effect on the performance of Max-sum. We mentioned above two optimization criteria that can be used (*max-contribution* and *max-coverage*). We demonstrate that symmetry affects both criteria in both aggregation functions.

When F_{Sum} is used in combination with the *max-contribution* criteria, then for each possible location for the neighboring agents, the target will calculate for the sensor its maximal contribution. In this case the

maximal contribution for the agent is when the other agents contribute the least possible (e.g., when using F_{Sum}, they are located as far as possible). Therefore, the agent would get the maximal possible contribution for every assignment within SR from the target regardless of how many other agents are in range. Hence, in many cases all neighboring agents will receive messages that incentivize them to move closer to the target, thus, generating clusters and avoiding exploration. On the other hand, if we use the *max-coverage* criteria, then in cases where there are more neighbors than required to cover the target, agents will be informed that regardless of their selection, the target is covered. The agents would all be indifferent in their decision whether to cover the target or not, and the result might be that not enough agents select positions within SR of the target, i.e. a lower coverage than required.

The symmetry drawback is even more acute in implementations using F_{PP}, in which symmetry may cause a reduction in performance both as a result of clustering (as in F_{Sum}), and due to lack of coordination in the deployment of agents surrounding a target. For each possible position of a sensor, the function-node will report the utility considering that all other agents are located such that the angle between their lines of sight is maximal. Thus, the agent will be indifferent between the possible locations available for it. Since all of the agents receive similar information, they select their position arbitrarily and this may result in low quality of the overall solution (a similar problem has been reported by Zivan and Peled (2012) for graph coloring problems).

We note that for an implementation using F_{Sum}, this form of symmetry does not lower the quality of the solution, since coverage is only determined by the distance from the target and not by the relative proximity. Fig. 3 demonstrates how symmetry reduces the effectiveness of Max-sum in the F_{PP} implementation of DCOP_MST. The scenario depicted includes two agents A_1 and A_2 within range of target T_1. They each have four assignments (possible locations) in their domain $a_{ij}, i \in 1,2, j \in 1,\ldots,4$. The joint credibility function used is F_{PP}, with $MA_{T_1} = 180°$. When considering the possible assignments of agent A_1, one notices that assignments a_{11} and a_{12} are in sensing range of T_1 (while the other two assignments are not) and for agent A2, assignments a_{21} and a_{22} are in sensing range of T_1. The table at the bottom of Fig. 3, presents the messages that the agents receive from the target (after constant reduction). The messages attribute the same maximal utility for all the assignments within range of the target because it considers a situation where the other agent is located in the opposite position. The result is that both agents are indifferent in choosing any assignment in range and would not necessarily choose opposite locations. If the agents use a random tie breaking technique, in 50% percent of the cases the chosen assignment would be suboptimal. This happens if both agents select value assignments within sensing range, which the angle between their line of sights to the target is 90°, each of them would bear a reduction of 25% of their credibility and the resulting coverage of the target would be 75%. The highest utility is derived when the other agent is located such that the angle between their line of sights to the target is 180°. As mentioned above, symmetry problems also exist when using F_{sum} but only in the form of clustering. In the example presented in Fig. 3, regardless of the optimization criteria we would have selected, the message would include a difference of 5 between the assignments within range and the assignments out of range for each of the agents. Thus, both agents would come within range and the target would be covered.

4.5.1. The ordered value propagation solution for the symmetry problem

The solution we propose to overcome the symmetry problem is inspired by the ordered value propagation (OVP) approach proposed by Zivan and Peled (2012). First, an order on all the nodes in the factor graph is selected (e.g., according to indices). Next, the algorithm is performed for l iterations (where l is the diameter), allowing nodes to send messages only to nodes which are "after" them according to this order (in the case of ordering by indices, sending messages only to agents with larger indices than their own). After l iterations in the selected direction, the order is reversed and messages are sent for the next l iterations only in the opposite direction (i.e., to agents with lower indices). On iterations in which value propagation is performed, variable nodes include in their messages to function-nodes their selected value assignments. Function-nodes select the best cost considering only the value assignments they received from their variable-node neighbors, which are ordered before them.

In contrast, our implementation of OVP is a local method for target/function nodes, and does not include a global order as in Zivan and Peled (2012).[2] Each target (function-node) orders its neighboring sensors (variable-nodes) according to the number of constraints they are involved in. When a function node generates a message to a variable-node, it considers all the assignment combinations of neighbors that are after that sensor in the order, and only the selected assignments of neighbors that precede it in the order. For example, assume target T has k neighbors ordered $s_1, s_2, \ldots, s_k$. The message for s_1 is calculated as in standard Max-sum. Following the generation of the message to s_1, T selects a position for s_1 which is the one with highest utility in the message produced. In the calculation of the message to s_2, only the position selected for s_1 is taken under consideration. Thus, when the message to s_k is generated, the positions for all the other $k-1$ neighbors were already determined by target T.

OVP breaks the inherent symmetry of Max-sum and serves a dual purpose. In the F_{Sum} implementation, it helps avoid clustering of agents around targets by allowing the first sensors that are positioned by the target to select the best position for covering it, while the rest of the sensors realize that the target is covered and can explore for other targets. In the F_{PP} implementation, it also assists in coordinating the agents efficiently around the target as they sequentially select positions according to the previous ones selected locations and thus avoid the symmetry problem.

4.6. Exploration methods for Max-sum_MST

The modifications described in Sections 4 and 4.5 to the Max-sum algorithm result in performance, which is superior to standard local search (see experiments presented in Section 5). However, it is outperformed by local search algorithms that include explicitly designed exploration methods (Zivan et al., 2015).

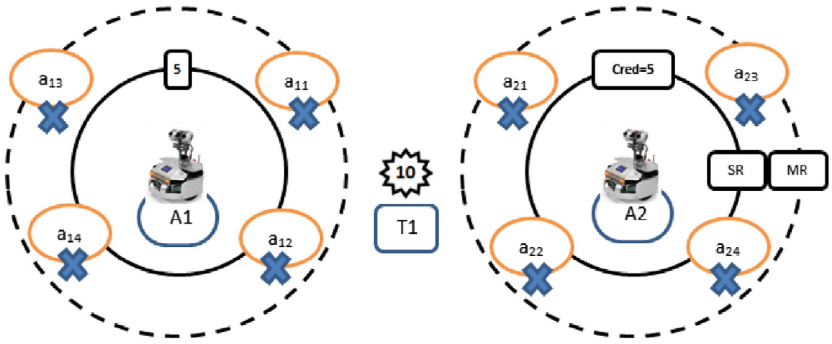
In contrast to local search algorithms, e.g., DSA, in which, when a target is completely covered, other agents would not consider covering it too, in Max-sum the highest utility that agents can receive, considering all possible locations of their neighboring sensors, is propagated by the targets for each possible location in its mobility range. This behavior creates clusters of agents around targets and prevents exploration.

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

4.6.1. Max-sum_PILR

In our first attempt to introduce exploration into Max-sum_MST, we attempted 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 is within c from the utility it would have derived if it would have selected the best possible position.

[2] Our empirical study included the evaluation of a number of global heuristics that were not found to outperform the local heuristic, which is obviously preferred in distributed settings.



Assignment	a_{i1}	a_{i2}	a_{i3}	a_{i4}
Values in messages from T_1	5	5	0	0

Fig. 3. An example of Max-sum_PP symmetry problem.

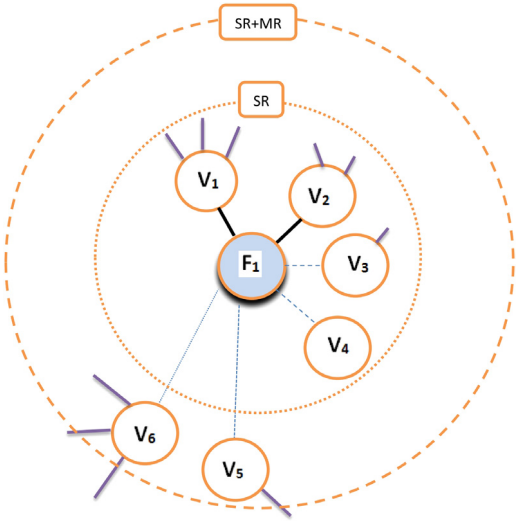


Fig. 4. An example of Max-sum_FMR.

4.6.2. 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 set of neighboring sensors of target t in FG_i, and by $Cred_{n(t)_i}$ the total credibility of $n(t)_i$. Denote by $r(t)_i$ a subset of $n(t)_i$ and by $Cred_{r(t)_i}$ the total credibility of $r(t)_i$. When there exists a subset $r(t)_i$ for which target t's importance is smaller than $Cred_{r(t)_i}$, t can select $r(t)_i$ neighbors for covering it and allow the other $n(t)_i - r(t)_i$ neighbors to perform exploration. We implement this by generating a new factor graph $\widehat{FG}_i$ in which each target t has at most $r(t)_i$ neighbors. This can be done distributively by having each target t remove the edges between it and $n(t)_i - r(t)_i$ of its neighbors. For homogeneous agents and targets, where $r(t)$ is the required number of sensors for covering target t, $r(t)$ is a constant number.

Fig. 4 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 joint credibility function used is the standard additive function F_{sum}. Therefore, three sensors are required at most for covering this target, hence, ($|r(t)_i| = 3$). 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 complexity 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 $|r(t)_i| - 1$. 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.[3]

The selection of the subset of 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 $|n(t)_i| > |r(t)|$. The heuristic is tuned with respect to the type of joint credibility function used:

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,
      i. If in F_{Sum} mode then remove the neighbor in $\hat{n}(t)_i$ that has the lowest degree from $n(t)$.

[3] 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.

ii. Else If in F_{PP} mode then remove the neighbor in $\hat{n}(t)_i$ that has the lowest contribution to the coverage of t.

When F_{Sum} is used, 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 SR. 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 SR, the selected sensor will need to move closer in order to be effective, thus, among the sensors outside of SR, it selects the least constrained ones. 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 implementations using F_{PP}, such a need to break ties between sensors with similar contribution is rare.

In the example presented in Fig. 4, 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 in $\bar{n}(T_1)_i$ (i.e., variable-nodes not within SR from target T_1), it has the highest degree. The second to be removed is V_5, which is also in $\bar{n}(T_1)_i$. Here, the heuristic is tuned according to the joint credibility function used. In both implementations the next variable-node to be removed is from set $\hat{n}(T_1)_i$ (variable-nodes within SR from T_1), since $\bar{n}(T_1)_i = \emptyset$. When F_{Sum} is used, all agents in $\hat{n}(T_1)_i$ have the same contribution and therefore, the sensor with the lowest degree would be removed, which is V_4 in this example. When F_{PP} is used, the agents' contributions vary and depend on their proximity to other agents. Therefore, in this example V_3 will be removed from the set as it is the sensor with the lowest contribution among the rest (it has the closest neighbors).

4.7. Tie breaking

The FMR method detaches the connections between targets and some of their neighboring agents. The objective of this detachment is to encourage these agents to explore for other targets, where their sensing is required. However, currently the agents are indifferent between staying in their current location and selecting new locations. This indifference is caused by the fact that no function nodes are sending messages specifying utilities for the different possible locations, and in the agents' view, they will get zero utility for any location they will select. The method used for breaking such ties has a dramatic effect on the performance of the algorithm proposed in this work.

Commonly, ties between utilities of different values in variable's domains are solved using preferences (unary constraints) of each agent over its value assignments, as suggested in Farinelli et al. (2008). The numerical values of these preferences are orders of magnitude smaller than the utilities derived from the standard constraints of the problem, thus, only in the case of ties they affect the selection of the assignment. We will denote this method by $Pref$. In cases of ties between many (or all) values in the agents' domains, the $Pref$ method will results in agents selecting over and over the same value assignment and preventing them from exploring. Instead, an agent can select an assignment randomly among the tied values that offer the highest utility. This method, which we termed $Rand$, allows the agents to continuously explore new positions and the targets that may be covered from these positions.

It is important to notice that $Rand$ is not used in standard Max-sum because in DCOP, all the problems constraints are known to the agents. Thus, there is no need to incentivize agents to explore. The objective of tie breaking is simply to coordinate the selection of the same solution among agents. In DCOP_MST on the other hand, agents are motivated to seek for locations from which they are more effective via the tie breaking method. While it enhances exploration, the $Rand$ method has a dichotomous effect in terms of the overall performance of the team. On the one hand, it assists agents to explore their surrounding and perhaps locate uncovered targets and improve their coverage. On the other, it intensifies the clustering effect. For the Max-sum_FMR algorithm, which prevents clustering, it is important and effective to add $Rand$ tie breaking to encourage the agents that are not selected for coverage by over-covered targets to explore for alternative locations. When used with $Pref$ tie breaking, FMR loses its effectiveness as agents are not motivated to explore.

5. Experimental evaluation of Max-sum_MST

In order to compare Max-sum_MST and its versions that include exploration with the DCOP_MST algorithms proposed in Zivan et al. (2015), 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 0 and 100. The methods for calculating the joint coverage of agents within the sensing range of a target are the F_{Sum} and F_{PP}. Max-sum algorithms ran 5 rounds of messages in every iteration. This number was found to produce best results.[4] All results depicted in this section are an average over 50 runs of the algorithm solving 50 different random problems in terms of the initial locations of the agents and the targets. In each experiment the specific values for $m, SR, MR, Cred_i, ER, |A|, |T|$, and F where chosen differently to demonstrate the algorithms' behavior in different scenarios.

5.1. Evaluation on problems where F_{Sum} is used

In this section we compare MST algorithms solving problems in which the F_{Sum} joint credibility function is used. The experimental setting included problems with 50 agents (sensors) and 20 targets that were randomly deployed in a 100 over 100 grid. Each target had $ER = 100$. $SR = 5$, $MR = 5$. The credibility variable in this set of 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.

Fig. 5 presents a comparison between Max-sum_MST, the standard local search DCOP algorithms that were adjusted to DCOP_MST in Zivan et al. (2015) and the PILR explorative algorithms. For each of the Max-sum versions we implemented two tie breaking methods $Rand$ and $Pref$ as explained in Section 4.7. 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 selected 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., $r(t) = 4$. The results demonstrate the advantage of Max-sum_MST with standard ($Pref$) tie-breaking, over standard DSA. Not only does it converge in a smaller number of iterations,[5] but its final result is also better than DSA. However, it is clearly inferior to DSA_PILR, which is combined with exploration methods. Nevertheless, when using random tie breaking ($Rand$), Max-sum outperforms DSA_PILR as well. In fact, all versions of Max-sum using standard tie-breaking performed similarly (we left only one line among them to avoid density). However,

[4] A smaller number of message rounds produced inferior results, while for a larger number we experienced loopy propagation behavior as described in Zivan and Peled (2012) and Chli and Winsper (2015), which again, produced lower quality solutions.

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

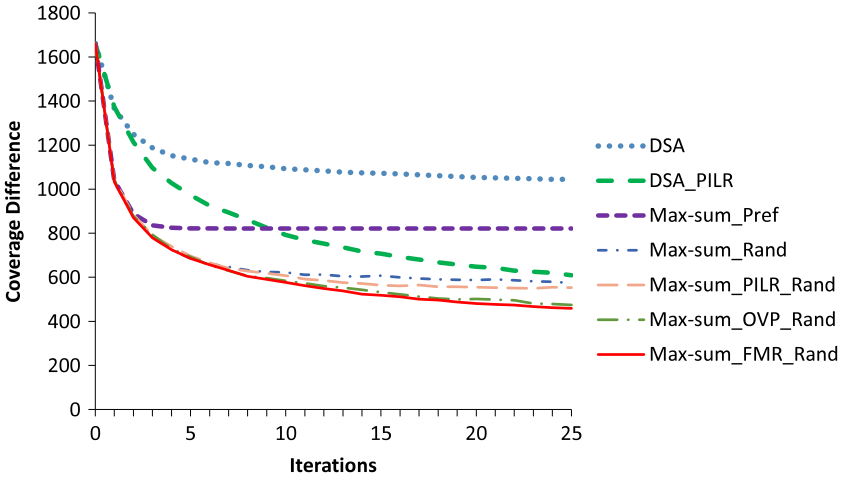


Fig. 5. Sum of coverage differences over all targets as a function of the number of iterations. $SR = 5$, $MR = 5$, F_{Sum}.

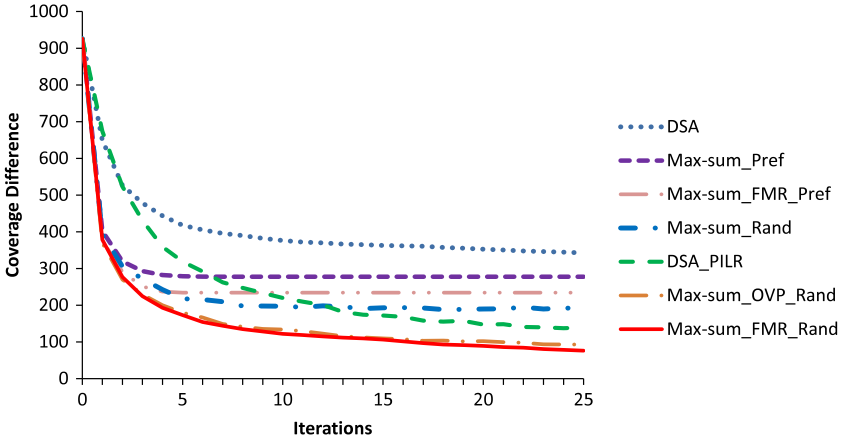


Fig. 6. Sum of coverage differences for all targets, as a function of the number of iterations. $SR = 10$, $MR = 5$, F_{Sum}.

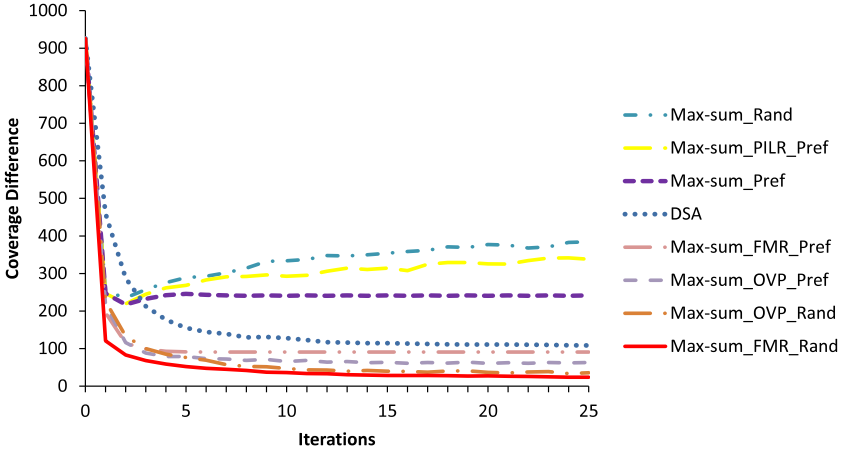


Fig. 7. Sum of coverage differences for all targets, as a function of the number of iterations. $SR = 10$, $MR = 10$, F_{Sum}.

the versions using random tie breaking outperformed DSA_PILR. Among them, the OVP and FMR versions performed slightly better than standard Max-Sum and Max-sum_PILR.

Figs. 6 and 7 present results for experiments in which the local environments of agents were larger. In Fig. 6, the ranges were $SR = 5$ and $MR = 10$. In this scenario there was a significant difference between the standard version of Max-sum using standard tie breaking and the FMR version. Both Max-sum_PILR versions produced similar results to standard Max-sum with random tie-breaking, therefore we omitted the lines from the graph to avoid density. In this scenario, the FMR and OVP

Table 1
The average maximal time in seconds required to complete an iteration as a function of different sensing and mobility ranges for the F_{Sum} implementation. Max-sum versions are shown with and without the THOP speedup.

Parameters		Algorithms			
MR	SR	DSA	Max-sum (w/o THOP)	Max-sum (THOP)	Max-sum_FMR
5	3	0.00	0.15	0.04	**0.05**
5	4	0.01	0.17	0.08	**0.07**
5	5	0.01	0.25	0.10	**0.08**
5	6	0.02	0.35	0.14	**0.09**
5	7	0.03	0.42	0.16	**0.11**
5	8	0.04	0.46	0.21	**0.13**
5	9	0.05	0.53	0.27	**0.16**
5	10	0.06	0.68	0.35	**0.20**
6	10	0.07	1.17	0.44	**0.24**
7	10	0.09	1.42	0.51	**0.28**
8	10	0.11	6.04	0.56	**0.31**
9	10	0.12	30.84	0.63	**0.34**
10	10	0.15	–	0.72	**0.38**

Table 2
The average maximal time in seconds required to complete an iteration as a function of different sensing and mobility ranges for the F_{PP} implementation.

Parameters		Algorithms		
MR	SR	DSA	Max-sum_MST	Max-sum_FMR
1	1	0.01	4.4	**0.7**
1	2	0.02	15.3	**1.0**
2	2	0.09	514.1	**2.1**

versions with random tie-breaking produced results with a significant advantage over standard Max-sum[6] This advantage is more apparent in Fig. 7. In fact, when the ranges grow there is a deterioration in the versions of Max-sum and Max-sum_PILR using random tie-breaking. This deterioration is due to the large local environments that allow agents to move towards the most attractive positions (the positions from which agents can derive maximal utility from) in the area. On the other hand, this phenomenon does not affect Max-sum_OVP and Max-sum_FMR that have an inherent mechanism to address the symmetry problem. Thus, in these versions, while some agents shift towards the most attractive positions in the area, others maintain coverage of targets in positions from which they derive less utility.

We refrain from adding error bars to the figures as they become overly cluttered. Yet, in all the experiments, the difference between the performance of Max-sum_FMR and DSA_PILR is statistically significant within a 5% confidence interval.

Table 1 presents a runtime comparison between the algorithms solving problems in which the F_{sum} joined utility function is used, on which it is possible to use the THOP method. 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 an iteration averaged over 50 random experiments for varying mobility and sensing ranges. The use of the THOP technique enables Max-sum to run with larger ranges without requiring exponential computation. The MR in the experiments was varied between 5 and 10 and the SR was varied between 3 and 10. The growth in runtime demonstrated in Table 1 is linear. While in DSA and DSA_PILR exhibit a smaller increase in runtime compared to Max-sum, it is apparent that with THOP, Max-sum and its explorative versions can run on large and dense problems and achieve good results in terms of coverage. Max-sum_FMR requires near linear runtime while producing the best results in terms of coverage quality.

While Max-sum_FMR and Max-sum with THOP manage to prevent the exponential runtime of standard Max-sum, the runtime of DSA is still significantly lower. However, we should keep in mind that DCOP_MST is used to represent robotic applications. In such applications, the movement time of the robots is usually much larger than computation time. Furthermore, there exists a trade-off between solution quality and computation time. As long as the runtime for computing the movements of robots is not extremely high (exponential as in standard Max-sum), the slowdown that results from Max-sum_FMR, seems to be worthwhile considering the quality of the resulting coverage.

5.2. Evaluation on problems where F_{PP} is used

In this section we compare MST algorithms solving problems in which the F_{PP} joint coverage function is used. The experiments included 10 agents with credibility 60 and 4 targets with importance 100, that were deployed randomly in a 20 over 20 grid. $SR = 2$, $MR = 2$.[7] The number of agents required to completely cover a target is 2 and therefore the minimal angle between agents that does not cause a utility reduction was $Ang_{min} = 180$. Table 2 presents the runtime comparison on problems in which the F_{pp} joint utility function is used. The exponential growth in runtime is apparent in standard Max-sum. However, Max-sum_FMR does not exhibit this exponential increase in runtime, as expected.

Fig. 8 presents a comparison between Max-sum_MST, its explorative version Max-sum_FMR, the standard local search DCOP algorithm DSA and its explorative version DSA_PILR, solving problems in which F_{PP} is used. We demonstrate the effect of using the OVP method and the random tie breaking. In order to avoid density we omitted some of the lines and error bars. DSA performed similar to Max-sum with preference tie breaking. The Max-sum_PILR versions performed similar to Max-sum_FMR_OVP with preferences tie breaking. The combination of the FMR and OVP methods with random tie breaking produced the best results. The difference between the performance of Max-sum_FMR_OVP_Rand and DSA_PILR is statistically significant within a 5% confidence interval. It is important to mention that while these results only slightly improve the results of Max-sum with OVP and random tie breaking, the FMR method insures that the computations are not exponential in the number of neighbors and therefore, results are obtained much faster in the versions that use FMR.

6. Conclusions

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

[6] Similar results were obtained for sensors with $SR = 10$ and $MR = 5$. They were omitted to avoid redundancy.

[7] The scenario selected in this section was much smaller due to the exponential runtime required by function-nodes in Max-sum.

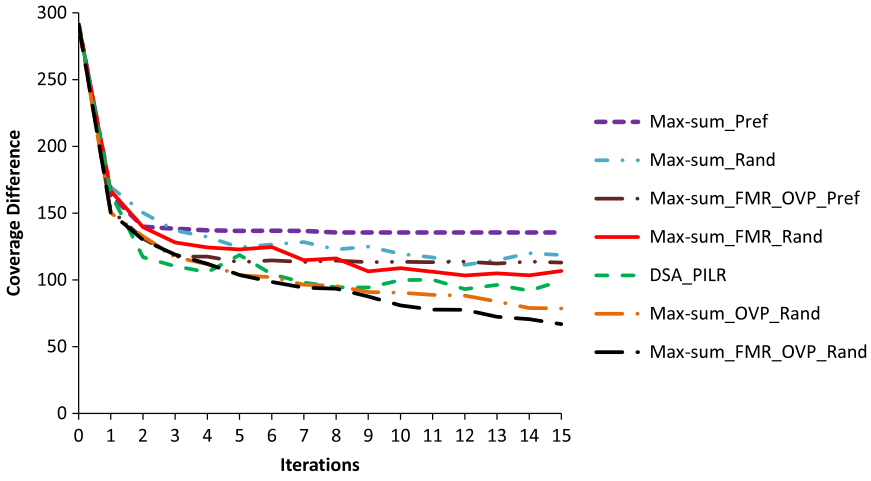


Fig. 8. Sum of coverage differences over all targets, as a function of the number of iterations. $SR = 2$, $MR = 2$ F_{PP}.

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.

Two scenarios were considered. The first allowed the use of the THOP method that makes the calculations performed by function-nodes in Max-sum efficient. The second included different utilities with respect to the specific angles between the sight lines of agents to targets. Thus, it did not allow the use of THOP and required exponential computation.

We demonstrated that Max-sum generates a symmetry problem when solving MST problems, in which agents are encouraged to cluster near targets, and demonstrated that these symmetries generated poor results in the combinatorial problems (that do not allow the use of THOP). We solved the symmetry problem by designing a local version of ordered value propagation.

Our empirical evaluation demonstrates that the combination of symmetry breaking and exploration improve immensely the performance of Max-sum when solving MST problems. Max-sum_FMR also avoids exponential computation even when solving combinatorial problems in which THOP cannot be used.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.engappai.2018.02.017.

References

Abramson, M., Chao, W., Mittu, R., 2005. Design and evaluation of distributed role allocation algorithms in open environments. In: The International Conference on Artificial Intelligence, pp. 565–571.

Aiello, F., Fortino, G., Guerrieri, A., Gravina, R., 2009. Maps: a mobile agent platform for wsns based on java sun spots, In: Proceedings of the ATSN.

Chli, M., Winsper, M., 2015. Using the max-sum algorithm for supply chain emergence in dynamic multiunit environments. IEEE Trans. Syst. Man Cybern. 45 (3), 422–435.

Das, S., Goswami, D., Chatterjee, S., Mukherjee, S., 2014. Stability and chaos analysis of a novel swarm dynamics with applications to multi-agent systems. Eng. Appl. Artif. Intell. 30, 189–198.

Erdem, U.M., Sclaroff, S., 2006. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. Comput. Vis. Image Underst. 103, 156–169.

Farinelli, A., Rogers, A., Jennings, N., 2013. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. J. Auton. Agents Multi-Agent Syst. 337–380.

Farinelli, A., Rogers, A., Petcu, A., Jennings, N.R., (2008) Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: AAMAS, pp. 639–646.

Fortino, G., Galzarano, S., 2013. On the development of mobile agent systems for wireless sensor networks: issues and solutions. Multiagent Syst. Appl. 185–215.

Hermelin, D., Segal, M., Yedidsion, H., 2017. Coordination of mobile mules via facility location strategies. In: International Conference on Practical Applications of Agents and Multi-Agent Systems. Springer, pp. 107–119.

Hollinger, G.A., Pereira, A.A., Binney, J., Somers, T., Sukhatme, G.S., 2016. Learning uncertainty in ocean current predictions for safe and reliable navigation of underwater vehicles. J. Field Robot. 33 (1), 47–66.

Hsieh, F.-S., 2009. Developing cooperation mechanism for multi-agent systems with Petri nets. Eng. Appl. Artif. Intell. 22 (4), 616–627.

Hu, H., Oyekan, J., Gu, D., 2011. A school of robotic fish for pollution detection in port. In: Liu, Y., Sun, D. (Eds.), Biologically Inspired Robotics. pp. 85–104.

Jain, M., Taylor, M.E., Yokoo, M., Tambe, M., 2009. DCOPs meet the real world: exploring unknown reward matrices with applications to mobile sensor networks. In: Proceedings of the International Joint Conference on Artificial Intelligence.

Kim, Y., Lesser, V.R., 2013. Improved max-sum algorithm for dcop with n-ary constraints. In: AAMAS.

La, H., Sheng, W., 2013. Distributed sensor fusion for scalar field mapping using mobile sensor networks. IEEE Trans. Cybern. 43 (2), 766–778.

Le, T., Fioretto, F., Yeoh, W., Son, T.C., Pontelli, E., 2016. Er-dcops: a framework for distributed constraint optimization with uncertainty in constraint utilities. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 606–614.

Leitão, P., Barbosa, J., Trentesaux, D., 2012. Bio-inspired multi-agent systems for reconfigurable manufacturing systems. Eng. Appl. Artif. Intell. 25 (5), 934–944.

Lesser, V., Ortiz Jr., C.L., Tambe, M., 2012. Distributed Sensor Networks: A Multiagent Perspective, Vol. 9. Springer Science & Business Media.

Li, S., Negenborn, R.R., Lodewijks, G., 2016. Distributed constraint optimization for addressing vessel rotation planning problems. Eng. Appl. Artif. Intell. 48, 159–172.

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

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

Maheswaran, R.T., Tambe, M., Bowring, E., Pearce, J.P., Varakantham, P., 2004b. Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1. IEEE Computer Society, pp. 310–317.

Meisels, A., 2008. Distributed Search by Constrained Agents: Algorithms, Performance, Communication. Springer Science & Business Media.

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

Nguyen, T., Yeoh, W., Lau, H.C., Zilberstein, S., Zhang, C., 2014. Decentralized multi-agent reinforcement learning in average-reward dynamic DCOPs. In: AAAI, pp. 1341–1342.

Nooner, S.L., Chadwick, W.W., 2009. Volcanic inflation measured in the caldera of axial seamount: Implications for magma supply and future eruptions. Geochem. GeoPhys. Geosyst. 10 (2).

Petcu, A., Faltings, B., 2005. Superstabilizing, fault-containing multi-agent combinatorial optimization. In: AAAI, pp. 449–454.

Pujol-Gonzalez, M., Cerquides, J., Meseguer, P., Rodríguez-Aguilar, J.A., Tambe, M., 2013. Engineering the decentralized coordination of uavs with limited communication range. Adv. Artif. Intell. 199–208.

Ramchurn, S.D., Farinelli, A., Macarthur, K.S., Jennings, N.R., 2010. Decentralized coordination in robocup rescue. Comput. J.

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

Rust, P., Picard, G., Ramparany, F., 2016. Using message-passing dcop algorithms to solve energy-efficient smart environment configuration problems. In: IJCAI, pp. 468–474.

Smith, R.N., Chao, Y., Li, P.P., Caron, D.A., Jones, B.H., Sukhatme, G.S., 2010. Planning and implementing trajectories for autonomous underwater vehicles to track evolving ocean processes based on predictions from a regional ocean model. Int. J. Robot. Res. 29 (12), 1475–1497.

Stranders, R., Farinelli, A., Rogers, A., Jennings, N.R., 2009. Decentralised coordination of mobile sensors using the max-sum algorithm. In: IJCAI, pp. 299–304.

Stranders, R., Tran-Thanh, L., Fave, F.M.D., Rogers, A., Jennings, N., 2012. Dcops and bandits: Exploration and exploitation in decentralised coordination. In: AAMAS, pp. 289–297.

Tarlow, D., Givoni, I., Zemel, R., 2010. Efficient message passing with high order potentials. In: AISTATS, Vol. 9, pp. 812–819.

Taylor, M.E., Jain, M., Jin, Y., Yokoo, M., Tambe, M., 2010. When should there be a "me" in "team"?: distributed multi-agent optimization under uncertainty. In: AAMAS, pp. 109–116.

Vargo, E., Bass, E., Cogill, R., 2013. Belief propagation for large-variable-domain optimization on factor graphs: an application to decentralized weather-radar coordination. IEEE Trans. Syst. Man Cybern. 43 (2), 460–466.

Vazquez, P., Feixas, M., Sbert, M., Heidrich, W., 2003. Automatic view selection using viewpoint entropy and its application to image based modelling. Comput. Graph. Forum 22, 689–700.

Voice, T., Stranders, R., Rogers, A., Jennings, N.R., 2010. A hybrid continuous max-sum algorithm for decentralised coordination. In: ECAI, pp. 176–182.

Wang, G., Cao, G., Berman, P., Laporta, T.F., 2003. Bidding protocol for deploying mobile sensors. In: IEEE ICNP, pp. 315–324.

Wang, Y., Cai, Z., Guo, G., Zhou, Y., 2007. Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Trans. Syst. Man Cybern 37 (3), 560–575.

Wang, Y., de Silva, C.W., 2008. A machine-learning approach to multi-robot coordination. Eng. Appl. Artif. Intell. 21 (3), 470–484.

Xiang, W., Lee, H., 2008. Ant colony intelligence in multi-agent dynamic manufacturing scheduling. Eng. Appl. Artif. Intell. 21 (1), 73–85.

Yedidsion, H., Zivan, R., Farinelli, A., 2014. Explorative max-sum for teams of mobile sensing agents. In: AAMAS, pp. 549–556.

Yeoh, W., Felner, A., Koenig, S., 2008. BnB-Adopt: An asynchronous branch-and-bound dcop algorithm. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2, International Foundation for Autonomous Agents and Multiagent Systems, pp. 591–598.

Yeoh, W., Sun, X., Koenig, S., 2009. Trading off solution quality for faster computation in dcop search algorithms. In: IJCAI.

Zhang, W., Xing, Z., Wang, G., Wittenburg, L., 2005. Distributed stochastic search and distributed breakout: properties, comparishon and applications to constraints optimization problems in sensor networks. Artificial Intelligence.

Zivan, R., 2008. Anytime local search for distributed constraint optimization. In: AAMAS, pp. 393–398.

Zivan, R., Okamoto, S., Peled, H., 2014. Explorative anytime local search for distributed constraint optimization. Artificial Intelligence 212, 1–26.

Zivan, R., Peled, H., 2012. Max/min-sum distributed constraint optimization through value propagation on an alternating dag. In: AAMAS, pp. 265–272.

Zivan, R., Yedidsion, H., Okamoto, S., Glinton, R., Sycara, K.P., 2015. Distributed constraint optimization for teams of mobile sensing agents. J. Auton. Agents Multi-Agent Syst. 29, 495–536.