CAMS: Collision Avoiding Max-Sum for Mobile Sensor Teams

Arseni Pertzovskiy
Ben-Gurion University of the Negev
Beer Sheva, Israel
arsenip@post.bgu.ac.il

Roie Zivan
Ben-Gurion University of the Negev
Beer Sheva, Israel
zivanr@bgu.ac.il

Noa Agmon
Bar-Ilan University
Ramat Gan, Israel
agmon@cs.biu.ac.il

ABSTRACT

Recent advances in technology have large teams of robots with limited computation and communication skills work together in order to achieve a common goal. Their personal actions need to contribute to the joint effort, however, they also must assure that they do not harm the efforts of the other members of the team, e.g., as a result of collisions. We focus on the distributed target coverage problem, in which the team must cooperate in order to maximize utility from sensed targets, while avoiding collisions with other agents. State of the art solutions focus on the distributed optimization of the coverage task in the team level, while neglecting to consider collision avoidance, which could have far reaching consequences on the overall performance. Therefore, we propose CAMS: a collision-avoiding version of the Max-sum algorithm, for solving problems including mobile sensors. In CAMS, a factor-graph that includes two types of constraints (represented by function-nodes) is being iteratively generated and solved. The first type represents the task-related requirements, and the second represents collision avoidance constraints. We prove that consistent beliefs are sent by target representing function-nodes during the run of the algorithm, and identify factor-graph structures on which CAMS is guaranteed to converge to an optimal (collision-free) solution. We present an empirical evaluation in extensive simulations, showing that CAMS produces high quality collision-free coverage also in large and complex scenarios. We further present evidence from experiments in a real multi-robot system that CAMS outperforms the state of the art in terms of convergence time.

KEYWORDS

Distributed Constraint Optimization Problems (DCOP); Mobile Sensor Teams; Max-sum Belief Propagation

ACM Reference Format:
Arseni Pertzovskiy, Roie Zivan, and Noa Agmon. 2023. CAMS: Collision Avoiding Max-Sum for Mobile Sensor Teams. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Some of the most challenging multi agent applications involve teams of mobile sensing agents that are required to acquire information in a given area. Examples for such applications are networks of sensors [14, 41], smart homes [24], and rescue teams in disaster areas [17]. The mobile agents reside on physical aerial or ground devices, thus they must avoid collisions. Moreover, the dynamic nature of the environments the agents operate in, as well as their commonly limited communication, computation and sensing capabilities, require that the decisions made by each agent are fast and short term. The ability of those local short-term decisions to result in an overall optimal group strategy is a fundamental challenge for multi-agent systems, and it is the focus of this paper. Specifically, we examine the problem of *distributed target covering* by a team of limited robots, in which the robots decide which targets to sense for yielding maximal group utility.

Distributed constraint optimization problems (DCOP) offer a framework that addresses some of the above challenges. As DCOPs are limited in representing dynamic events, an extension of the DCOP framework, DCOP_MST (Mobile Sensor Team), along with local search algorithms, were proposed by Zivan et al. [41]. A later study has shown that an incomplete inference algorithm, Max-sum [2, 7, 9], produces better results when used in an iterative process for solving DCOP_MST, where iterative instances of the current representation of the problem are solved distributively and allow the agents to select the next joint move [37]. The Max-sum algorithm has been the subject of intensive study in DCOP solving research, and has been applied to many realistic applications [8, 22, 24], among those for solving DCOP_MST, by the Max-sum_MST algorithm [41]. Previous studies that investigated the performance of Max-sum when solving standard DCOPs (e.g., [5, 40]) reported that Max-sum (without the addition of function-node splitting) oscillates for thousands of iterations, whether it finally converges or not. The nature of this phenomenon has remained an open question. In contrast, when applied to DCOP_MST, Max-sum converges instantly [37].

While previous DCOP-based work offer solutions that can be successfully applied to target covering, they neglect to consider in their optimization criteria one critical aspect of the problem: collision avoidance. Since the team members act physically in the environment, they cannot collide with each other. As seen vastly in robotics research, and in the research area of Multi-Agent Path Finding (MAPF), accounting for collisions between physical agents is extremely challenging, and has far-reaching consequences on the performance of the system [13, 29]. However, as opposed to MAPF where the goal is to globally create collision-free paths optimizing some joint path-length criterion, or collision-avoiding in robotics research that focus on generating locally safe trajectories, here we are interested in *local* decision making for *target-covering optimization*, where collision-avoidance being an additional important constraint.

Therefore this paper addresses the two above important challenges that arise when using incomplete distributed inference algorithms for solving dynamic mobile sensing team problems. The first is the enigma related to the fast convergence of Max-sum when applied to DCOP_MST. We prove that the convergence results from

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

the special structure of MST problems and that the messages content sent by the nodes representing the constraints in Max-sum will be fixed when solving DCOP_MST.

The second challenge that we address is the need to avoid collisions while solving locally optimization problems, modeled as DCOPs. We propose a collision-avoiding version of Max-sum, called CAMS. As in standard Max-sum, the problem in CAMS is represented by a factor-graph, which is a bipartite graph including nodes representing variables and functions (constraints). In addition to the standard optimization using Max-sum, CAMS adds a new type of function-nodes to the factor-graphs, which represent the constraints of locations that agents can choose to move to.

We identify structures of factor-graphs on which CAMS is guaranteed to converge to the optimal (collision avoiding) solution. We prove that the added complexity of CAMS with respect to Max-sum_MST is small. We show empirically in extensive simulations and using real robotic systems that even for complex cases (e.g., in dense and dynamic environments), CAMS converges efficiently to high quality collision-free solutions.

2 RELATED WORK

The DCOP model has been widely used for representing and solving coordination problems related to sensor networks [8, 18] and mobile sensor networks [30, 32, 41]. To the best of our knowledge, *none of these studies addressed the possibility of collisions between mobile sensors*. Nevertheless, an attempt to use the DCOP model and algorithms in order to avoid collisions of ships was presented in [11], proposing the distributed stochastic search algorithm (DSSA) for preventing ships from selecting colliding routes. We compare this version and our proposed algorithm with a collision avoiding DSA (CADSA) algorithm in our empirical study.

DCOPs are traditionally associated with problems in discrete settings [30, 32, 39, 41], including DCOP_MST. Attempts to investigate the modelling and solving of DCOPs in continuous domains (e.g., [12, 26, 33]), raise challenges concerning, among others, the type of continuous utility/cost functions and the impact of these types on the complexity of the problem. While the importance of generalizing DCOPs to continuous domains is noted, this paper follows the common discrete modeling, associated also with DCOP_MSTs (which is the baseline for this work).

Different aspects of the Max-sum algorithm have been examined in the literature, focusing on the algorithm's convergence guarantees [23, 39, 40], evaluation in realistic applications [22] and computational complexity [16, 17]. Our work contributes to this ongoing effort by extending the applicability of Max-sum to teams of mobile sensing agents.

While Max-sum has been shown to efficiently solve DCOPs and has been used to coordinate sensors' movements, one of its major drawbacks is the run-time required for function-nodes to produce messages, which is exponential in the arity of the constraint that the function-node represents. Multiple attempts to overcome this drawback were published in the last decade. Some of them, including the methods proposed by [17, 21], were implemented in the Max-sum version that was proposed for solving DCOP_MST in [36]. Others, which were proposed recently, make an immense reduction of the computational cost in such scenarios [3, 15]. In our work, we prove that such exponential computation is not required, and therefore these methods, which are most useful in standard scenarios, are less relevant.

Creating collision-free paths is the main objective of Multi-Agent Path Finding (MAPF) [1, 28, 29], which focus on creating paths for n agents on their way to their targets while avoiding spatial conflict between the agents, and optimizing some global criteria, usually minimizing the total travel distances or minimizing the makespan. Although MAPF algorithms and CAMS concentrate on collision avoidance, MAPF's difference from our problem is twofold: (1) The main objectives. While we care to optimize target covering as a cooperative effort accounting for collision avoidance as an additional constraint, MAPF focus on agents' paths to targets. (2) The basic model and means to solve the problem. CAMS solves DCOP_MST, an inherently-dynamic *distributed* constrained optimization problem requiring the agents to have no global knowledge of the world, and MAPF is a centralized problem, solved most commonly by centralized search-based methods. Therefore solutions to MAPF problems cannot be applied in our setting. Note that distributed MAPF was recently mentioned as one of the open challenges in MAPF [25]. Although there have been attempts to provide solutions to this problem (e.g., [20]), those still remain irrelevant for solving DCOP_MSTs.

Finally, collision avoidance, being one of the fundamental requirements from a robotic system, is vastly explored in multi-robot systems [13]. The main focus in *decentralized* collision avoidance methods is on providing means for locally preventing collisions by considering the other robots as mobile obstacles, or suggesting local coordination schemes yielding collision-free paths [6, 27, 31]. In our case collision avoidance is intertwined with the target covering task, necessitating the creation of a method that considers both for yielding an optimal team behavior, which is the essence of CAMS.

3 BACKGROUND

The DCOP model is commonly used for representing and solving coordination problems related to sensor networks [8, 18] and mobile sensor networks [30, 32, 41]. To the best of our knowledge, none of these studies addressed the possibility of collisions between mobile sensors. A vast amount of research has been invested in recent years in modelling and solving multi agent path finding problems (MAPF) [1, 28]. MAPF considers scenarios where a strong computing system (mostly centralized) is able to compute paths for all agents from their start position to their goal states while avoiding collisions. This is in contrast to scenarios on which we focus in this study, where the team of sensors is composed of entities with low computing and sensing abilities, that only compute a small number of steps ahead. An attempt to use the DCOP model and algorithms for collision avoidance of ships was presented in [11], where the distributed stochastic search algorithm (DSSA) was used in order to prevent ships from selecting colliding routes. Distributed stochastic algorithms (DSA) are synchronous local search algorithms in which agents hold assignments and make greedy attempts to improve them, subject to a stochastic replacement decision. A number of versions of DSA were found to be inferior to Max-sum_MST

in [36, 37]. Nevertheless, we compare the performance of our proposed algorithm with DSSA and a collision avoiding version of DSA (CADSA) in our empirical study.

In this section we provide the necessary background on Max-sum, the DCOP_MST model and Max-sum_MST. [1]

3.1 The DCOP_MST Model

DCOP_MST includes agents $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ physically situated in the environment, which is modeled as a metric space. Each agent A_i controls one variable, denoted by cp_i, that represents its current position. Time is discretized into an indeterminate series of time-steps, and the maximum distance A_i can travel in a single time step is defined by its *mobility range*, mr_i. Therefore, the domain of cp_i contains all locations within mr_i from it; consequently, once the agent moves, the content of its variable's domain changes. A change in the content of some variables' domains can induce a constraint change. The agents have limited heterogeneous sensing ranges, where 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 as their *credibility*. The credibility of agent A_i is denoted by $cred_i \in \mathbb{R}^+$, with higher values indicating better sensing abilities. Targets $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ 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. Thus, a target $T_j \in \mathcal{T}$ is a point p with $ER(p) > 0$.

Agents which their current position is within sensing range of target T_j are said to *cover* it and the *remaining coverage requirement* cr_j, is $ER(T_j)$ diminished by the joint credibility of the agents currently covering the target, with a minimum value of zero. The coverage of sensors aiming to apply to the environmental requirement of a target is not accumulated. Thus, if a target requires the coverage of more than one sensor, they must simultaneously place themselves in sensing range from it. Denoting the set of agents within sensing range of a point p by $sr(p)$, this is formalized as $cr(p) = \max\{0, ER(p) \ominus F(sr(p))\}$, where F is the joint credibility function that combines the credibility of neighboring agents and $\ominus : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$ is a binary operator that decreases the environmental requirement by the joint credibility. For simplicity we will assume that $F(sr(p)) = \sum_{A_i \in sr_{(p)}} cred_i$ and that $\ominus$ is a standard subtraction [37]. The global goal of the agents is to position themselves so to minimize $\sum_{T_j \in \mathcal{T}} cr(T_j)$. Such a minimization problem is NP-hard [34].

3.2 Distributed Stochastic Algorithm (DSA)

The distributed stochastic algorithm (DSA) is a simple synchronous search algorithm for solving DCOPs [38], which was applied to DCOP_MST, extended to avoid collisions, and which we use as a benchmark for comparison. In DSA, after an initial step in which agents select a value assignment for their variable (randomly according to [38]), agents perform a sequence of steps until some termination condition is met. In each step, an agent sends its value assignment to its neighbors and receives theirs. Versions of DSA differ in the stochastic method agents use to decide on whether to replace their value assignments. The algorithm uses a stochastic parameter $0 < p \leq 1$ in order to make this decision. If an agent in DSA cannot improve its current state (or keep the same cost, depending on the version used) by replacing its current value assignment, it does not replace it. Otherwise, it replaces its value assignment with probability p.

3.3 Convergence Properties

Belief propagation (in general and specifically Max-sum) converges in linear time to an optimal solution when the problem's corresponding factor-graph is acyclic (i.e., have a tree structured factor-graph) [19]. For a single-cycle factor-graph, we know that if belief propagation converges, then it is to an optimal solution [10, 35]. Moreover, when the algorithm does not converge, it periodically changes its set of assignments. In order to explain this behavior, Forney et al. [10] show the similarity of the performance of the algorithm on a cycle to its performance on a chain, whose nodes are similar to the nodes in the cycle, but whose length is equal to the number of iterations performed by the algorithm. One can consider a sequence of messages starting at the first node of the chain and heading towards its other end. Each message carries beliefs accumulated from utilities added by function-nodes. Each function-node adds a utility to each belief, which is the constraint value of a pair of value assignments to its neighboring variable-nodes. Each such sequence of utility accumulation (route) must at some point become periodic, and the maximal belief would be generated by the maximal periodic route. If this periodic route is consistent (i.e., the set of assignments implied by the utilities contain a single value assignment for each variable), then the algorithm converges. Otherwise, it does not.

We denote by a *path* the sequence of entries in the utility tables of the function-nodes along the route, which are accumulated in order to generate a belief.

3.4 Run-Time Complexity Analysis

The overhead in run-time complexity of CAMS (in comparison to Max-sum_MST) is negligible, since the additional function-nodes representing unary and binary constraints require at most 2^2 utility comparisons for each message produced (therefore there was no need to use recent published methods for reducing the computation of function-nodes in Max-sum [3, 15]). On the other hand, while target function-nodes may have more than two neighbors, we proved in Theorem 5.1 that the calculation is redundant, and thus, there is no need for these function-nodes to perform exponential computation. The computation performed by a variable-node in each step of the algorithm is the addition of the vectors received from its neighbors, for each message it sends. Thus, for an agent performing the computation of a single variable-node with k neighbors, the run-time complexity in each step of the algorithm is $k2(k-1) = O(k^2)$.

3.5 Applying Max-sum to DCOP_MST

Max-sum_MST, the Max-sum version applied to DCOP_MST, implements an iterative process in which in each iteration the

[1] For lack of space we moved the description of the standard DCOP model, Max-sum algorithm and additional proofs to the supplementary material, that can be found in https://u.cs.biu.ac.il/ agmon/CAMS-AAMAS23-Sup.pdf.

agents construct a factor-graph based on their current locations, run the Max-sum algorithm for a number of steps, and move according to the solution provided by the algorithm. In the next iteration, a new factor-graph is generated considering the new locations of the agents [37].

A message sent from function-node F representing target $T \in \mathcal{T}$ to a variable-node X, $R_{F\to X}$, includes two utilities, one for locations from which the sensor covers the target, and one for locations from which it does not (as in fast Max-sum [17]). A factor-graph in Max-sum_MST is generated using the function meta reasoning (FMR) method [37]. It is used when there are more neighbors than required for covering a target in order to avoid symmetry (i.e., prevent a situation in which all neighboring agents decide to cover some target, or alternatively, all decide not to cover it). 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 $T \in \mathcal{T}$ in FG^i. Denote by $r(T)^i$ a subset of $n(T)^i$ and let $cred_{r(T)^i} = \sum_{A_j \in r(T)^i} cred_i$. The function-node F representing target T selects the minimal subset $r(T)^i$ for which $ER(T) \leq cred_{r(T)^i}$, and removes the edges connecting it in the factor-graph to the sensors in $n(T)^i \setminus r(T)^i$. We will denote this set (the set of neighbors of T resulting from the FMR procedure in iteration i) by $N(T)^i$.

FMR is a first step to avoid symmetry, but it is not enough: If the accumulated credibility of target T's neighbors $N(T)^i$ is larger than its environmental requirements $ER(T)$, there is still a need to break symmetry in order to avoid sending distorted requirements to its neighbors (either high for all neighbors or low for all of them). Yedidsion et al. [37] suggested an ordered value propagation (OVP) approach, in which the neighbors are ordered and the utility for the last neighbor in the order is reduced. We propose a more balanced approach in the following section.

4 COLLISION AVOIDING MAX-SUM (CAMS)

Collisions among mobile sensors may result in damaging the sensors, execution delay, or even the inability to perform the coverage task. Thus, we propose Collision Avoiding Max-sum (CAMS) that allows the agents to select the deployment that maximizes coverage, while avoiding collisions. This is achieved by adding to the factor-graphs that are generated in each iteration of Max-sum_MST (before each movement of the agents) function-nodes representing locations that the agents can move to. Each such function-node can either represent a location to which only one agent can decide to move, or locations to which two agents can move.

In more details, in a factor-graph generated in CAMS there are three types of function-nodes:
1) FT_j, a function-node representing a target T_j. The neighbors of the function are selected using FMR. However, instead of performing ordered value propagation the targets adjust the utilities sent to their neighboring agents in a more balanced manner. Let u_{ij}^{cov} denote the utility sent to neighbor A_i by target FT_j for covering it. The value of u_{ij}^{cov} is defined as follows:

$$u_{ij}^{cov} = \begin{cases} ER_{FT_j}, \textbf{ if } ER_{FT_j} < cred_i \\ cred_i - \max\{0, \frac{\sum_{i' \in N(FT_j)} cred_{i'} - ER_{FT_j}}{|N(FT_j)|}\}, \textbf{ otherwise} \end{cases}$$

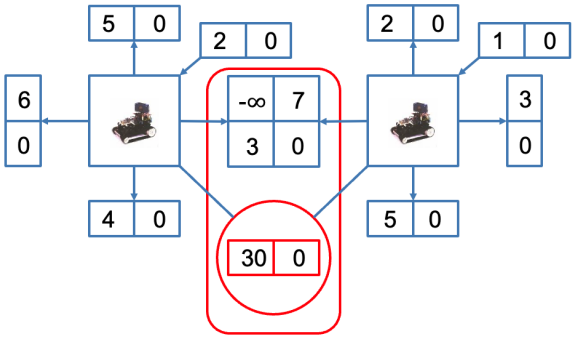


Figure 1: A factor-graph generated in CAMS.

We refer to this calculation of the neighbors coverage utilities as *balanced utility adaptation* (BUA).
2)$FL_{(i)}$, a function-node representing a location to which only one agent can move. The corresponding constraint is unary. A random positive utility is selected for the option that A_i selects this location (selected from the same range as the random utilities selected for the binary constraints described next) and zero for not selecting this location.
3)$FL_{(i,e)}$, a function-node representing a location l to which mobile sensors A_i and A_e can move in this iteration. The utility for both mobile sensors for not selecting location l is zero, for both selecting l is $-\infty$ and for both options in which only one of them selects l, a random utility is selected from a range of numbers that is much smaller than ER_{FT_j}[2]. In the case where one of the mobile sensors is located in l (without loss of generality, assume this is A_i), then the option that A_e moves to l and A_i moves to the current location of A_e is also excluded by utility -∞, and thus, edge constraints for avoiding collisions are enforced. We emphasize that although there may exist scenarios in which more than two mobile sensors can move to the same location, FL is defined as a binary constraint, and thus, if there are $k > 2$ mobile sensors that can select the same location, there will be an FL for each pair of these k mobile sensors.

Figure 1 presents an example of a factor-graph generated in some iteration of CAMS. It includes two mobile sensors, each with four possible locations to move to, and the option to stay in their current location. All function-nodes representing locations to which only one mobile sensor can move are of the second type. While the domain of each mobile sensor's current position variable includes five values (representing the possible locations it can select), only for the selection of the location represented by the function-node the utility is positive, and for all other locations it is zero. The middle location to which both mobile sensors can move, is represented by a function-node of the third type. It includes four options, one for both mobile sensors not selecting this location (zero utility), one for both selecting this location (minus infinity) and two with positive utilities for the cases that only one mobile sensor selects this location. The target is represented by a function-node of the first type. Its coverage requirement is 200, while the credibility of each mobile sensor is 70, which is the utility they derive for

[2]Random numbers are selected to avoid ties between desired options, as in [9]

	$Comb_1$	$Comb_2$	...	$Comb_{2^n}$
no_cover	u_1	u_2	...	u_{2^n}
$cover$	$u_1 + u^{cov}_{A,T}$	$u_2 + u^{cov}_{A,T}$	...	$u_{2^n} + u^{cov}_{A,T}$

Table 1: Function-node message generation.

covering the target. In this example, covering the target is only possible from the middle location that both mobile sensors can move to. However, if they both move to this location they collide.

5 PROPERTIES OF CAMS

In this section we discuss properties of CAMS that affect its convergence and its run-time analysis.

5.1 Messages sent by Target Representing Function Nodes

In order to identify the properties of the factor-graphs on which CAMS is guaranteed to converge to the optimal (collision-free) solution and to analyze its run-time properties, we aim to prove that function-nodes in Max-sum_MST (when implementing FMR, and BUA or OVP) do not change the content of the messages they send throughout the algorithm run. This property of Max-sum_MST has not been reported, nor proven, in previous studies.

We use in our proof a table (as depicted in Table 1), which represents the calculations performed by a target representing function-node FT, when generating a message to be sent to mobile sensor A. This table includes *two* rows, one for positions from which A covers the target and one for positions from which it does not. The columns represent combinations of position selections of all neighboring mobile sensors that can either select a position from which they cover the target or a position from which they do not. Thus, if there are n such neighbors (not including A), the number of columns will be 2^n. Each entry in the table includes the sum of the utility derived from the coverage, which the mobile sensors that selected a covering position in this column provided, and the relevant beliefs included in the messages received from the target's neighbors in the previous iteration.

Theorem 5.1. *In every iteration of CAMS and of Max-sum_MST, each target representing function-node will send to its neighbors in every step of the Max-sum algorithm, a message including zero utility for not covering this target and $u^{cov}_{A,T}$ for covering it (here, T is the target and A is the neighbor to which the message is sent).*

Proof: Table 1 demonstrates the calculation performed by function-node F_T, representing target T, when generating a message to be sent to its neighboring mobile sensor A. The message will include two utilities (beliefs), one for covering the target and one for not. As stated above, each of the columns represents a combination of positions selected by the target's other neighbors. The target selects the largest utility in each row to be included in the message it sends to A_i, one for not covering the target and the other for covering it. For each column j, the utilities are u_j for the first (non covering) row, and $u_j + u^{cov}_{A,T}$ for the row representing a covering position of A. Thus, if for some column j' and for all other columns $j \neq j'$, $u_{j'} > u_j$, then the message will include $u_{j'}$ for not covering the target and $u_{j'} + u^{cov}_{A,T}$ for covering it. Thus, after the reduction of $\alpha = u_{j'}$ the message sent will include $\langle 0, u^{cov}_{A,T} \rangle$. □

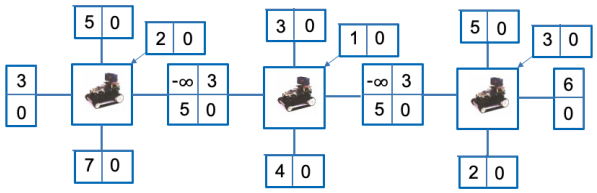


Figure 2: A tree structured factor-graph including only location representing function-nodes.

The significance of Theorem 5.1 is that it explains previously published evidence regarding the instant convergence of Max-sum_MST (e.g., [37]) in contrast to Max-sum's behavior on other benchmarks [2, 5].

5.2 Convergence Properties

We start the discussion by identifying factor-graph structures on which CAMS is guaranteed to converge to the optimal solution, and then we establish a more general property.

Lemma 5.2. *In any iteration of CAMS, if the factor-graph includes no cycles, and a collision-free solution exists, the algorithm will converge to the optimal collision-free solution in a linear number of steps.*

Proof: The factor-graph representation of this scenario has a tree structure and thus, Max-sum will converge in a linear number of steps to an optimal solution [19]. This optimal solution[3] cannot include the selection of the same location by two or more agents, since the utility of such a mutual selection is $-\infty$. □

This proof is immediate, given the properties of belief propagation as established in [19]. We mention it just to indicate that the hard constraint function-nodes do not interfere with this property, as long as a solution that does not violate hard constraints exists. Note that if the current position of the agents is collision free, then indeed such a solution exists, since the agents can choose to stay in their current locations.

Figure 2 presents an example of a tree-structured factor-graph including only location representing function-nodes. Note that additional function-nodes of the second type do not change the tree structure of the graph. In the optimal solution, the left mobile sensor moves down, the middle one moves to the left and the one on the right moves to the right. The (optimal) utility derived from this solution is 18.

Tree-structured graphs include all scenarios in which adjacent mobile sensors have only one location that they both can move to. However, when there are more than one such location, or when more than two mobile sensors can move to the same location, the factor-graph includes a cycle (See examples for two such scenarios in Figure 3).

Lemma 5.3. *In any iteration of CAMS, if the factor-graph includes a single cycle with two or more location representing function-nodes of*

[3]Here by optimal solution we mean the collision-free solution that provides the smallest remaining coverage in this iteration.

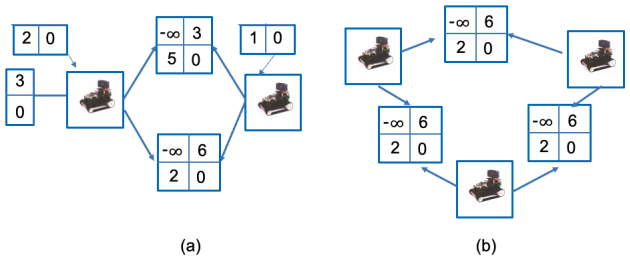


Figure 3: Single cycle factor-graphs including: (a) two mobile sensors and (b) three mobile sensors.

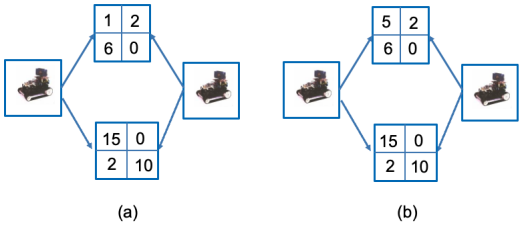


Figure 4: A single cycle factor-graph with two four entry function nodes.

the third type $FL_{(i,j)}$, and a collision-free solution exists, the algorithm converges to a collision-free optimal solution in a pseudo linear number of steps.

Proof: The factor-graph representation of such scenarios includes a single cycle with at least two $FL_{(i,j)}$ function-nodes. This type of function-node has four entries in the utility table, which one of them includes minus infinity utility. According to [10], when belief propagation is applied to a single cycle graph, it converges to the optimal solution if and only if the optimal (maximal in our case) repeated path is consistent. The only way to generate a maximal inconsistent path in such cycles including two or more function-nodes with four entry utility tables, is when the maximal path visits alternately entries of opposing directed diagonals in the utility tables (see proof in the supplementary material). Since in all utility tables the entry representing the movement of both mobile sensors to the represented location is equal to minus infinity, it is impossible to generate a maximal path including opposing directed diagonals. Thus, the maximal path must be consistent. The number of steps depends on the constant utilities sent by the unary function-node neighbors, which are not included in the cycle. If the difference between these utilities is negligible, the time for convergence is linear, i.e., in the order of the size of the cycle. □

In the proof above we stated that in a single cycle graph including two or more function-nodes representing binary constraints with four entries in their utility table, the maximal path can be inconsistent if and only if this path visits entries that are part of opposite direction diagonals. While the complete proof of this statement appears in our supplementary material, the following description provides intuition to its correctness. Consider such a cycle (as depicted in Figure 4 (a)). Assume each agent can select value assignments v_1 or v_2. Thus, the four entries represent the pairs of assignments $\langle v_1, v_1\rangle$ on the top left, $\langle v_2, v_1\rangle$ on the bottom left, $\langle v_1, v_2\rangle$ on the top right and $\langle v_2, v_2\rangle$ on the bottom right. Without loss of generality, we will follow a clockwise path that starts with the top left entry in the bottom utility table in the example depicted in Figure 4 (a). A consistent path will have both agents take v_1, i.e., the assignment will include $\langle v_1, v_1\rangle$ and the path will visit the left top entries in both utility tables until the algorithm terminates resulting in the path $15 \rightarrow 1 \rightarrow 15 \rightarrow 1 \ldots$. On the other hand, an inconsistent path includes shifts of assignments for each variable. However, since it is a path of a route, this shift is done alternately by the agents. Thus, after visiting $\langle v_1, v_1\rangle$ in the bottom utility table, the path shifts to $\langle v_1, v_2\rangle$ in the top utility table, then it shifts to $\langle v_2, v_2\rangle$ in the bottom utility table, to $\langle v_2, v_1\rangle$ in the top utility table and back to $\langle v_1, v_1\rangle$ in the bottom. The path visited is $15 \rightarrow 2 \rightarrow 10 \rightarrow 6 \rightarrow 15 \ldots$. One of these two passes is maximal (since the accumulated utility of all other consistent and inconsistent paths is much lower). In order to compare we need to normalize by length. The inconsistent path visits four entries that their sum is 33. The consistent path visits two entries, that their accumulated utility is 16. In order to normalize we will visit this path twice and get a utility of 32. Thus, the maximal path here is the inconsistent path. However, in the example depicted in Figure 4 (b) we increased the utility in the entry for $\langle v_1, v_1\rangle$ (top left) in the top utility table from 1 to 5. Now, the accumulated normalized utility of the maximal consistent path is 40, while the maximal accumulated utility of an inconsistent path remains 33. Thus, when solving the factor-graph depicted in Figure 4 (b), Max-sum would converge to the solution $\langle v_1, v_1\rangle$, while when solving the factor-graph depicted in Figure 4 (a) it would not converge.

A similar observation to the one we stated following Lemma 5.2 can be noted here as well. Additional unary constraints (function-nodes of the second type) would not affect the correctness of Lemma 5.3, since regardless of their content, they cannot overcome the minus infinity utilities that prevents an inconsistent maximal path.

Note that factor-graphs similar to the ones depicted in Figure 3(a) are generated when two mobile sensors can both move to two different locations, and that factor-graphs similar to the one depicted in Figure 3(b) are generated when three mobile sensors can move to a single location. When more than three mobile sensors can move to a single location, the representing factor-graph will include more than one cycle.

In order to state the final Theorem in this section, we introduce the following definition: The *Underlying location factor-graph* (ULFG) is the factor-graph that is generated in an iteration of CAMS, after removing all target representing function-nodes and all the edges connecting them to the mobile sensor variable nodes. Thus, all the function-nodes in a ULFG represent locations.

Theorem 5.4. *In any iteration of CAMS in which a factor-graph G is being solved by Max-sum, if the ULFG of G is tree structured or includes a single cycle, then Max-sum is guaranteed to converge on G to the optimal solution (collision-free, if such a solution exists) in a pseudo linear number of steps.*

Proof: According to Theorem 5.1, the target representing function-nodes in G constantly send the same messages. Thus, they act as if they are unary constraints (function-nodes with a single variable-node neighbor) and are not affected by the messages sent to them. According to [10, 39], Max-sum will converge if and only if the maximal path is consistent. As stated in the observations following Lemmas 5.2 and 5.3, unary constraints cannot change the fact that the maximal path in these graphs cannot be inconsistent. □

6 EMPIRICAL EVALUATION

In order to evaluate the performance of CAMS, we designed two types of experiments. The first set of experiments was performed in software simulation, implemented in Python[4], allowing for rigorous evaluation of CAMS compared to existing solutions. In the second set of experiments, we have fully implemented CAMS on a physical multi-robot system that included 3 Hamster robots [4]. We report here a subset of the results (identical trends were seen with other parameters, as well).

6.1 Evaluation in Simulation

The software simulation included two types of environments: grid, and random graphs. The obstacle-free grid contained $50X50$ cells (locations), and the random graphs contained 625 nodes, where each node had a random number of neighbors selected in the range $[2, 7]$. The graph environment represents a more complex setting in terms of coverage and collision constraints.

The number of targets was set to 20 and 10 for the grid and graph environments, respectively, and they were randomly positioned. The number of mobile sensors was 30 and 15, respectively, and they were also positioned randomly such that every location of the graph included at most one mobile sensor. For each environment, we tested the algorithms in both static and dynamic settings. In the static settings the number of targets and their location was constant, while in the dynamic setting new targets were added every t iterations (we used $t = 20$ for the grid, and $t = 10$ for the graph). The ER values of all targets were 100, the sensing range all sensors, sr_i, was set to 1 (its adjacent locations), and the credibility of each sensor was randomly selected between 25 and 50. The positive utilities of location function-nodes were selected randomly from the range $[10^{-10}, 10^{-5}]$. In each iteration of the algorithm, each mobile sensor could either move to one of the adjacent locations or stay in its current location.

Each experiment was executed 50 times, and we report the average remaining coverage and accumulated number of collisions obtained by each algorithm solving these scenarios, in each execution of the experiment.

We compared CAMS with six other algorithms: **(1)** Max-sum_MST (including FMR and OVP), in which the mobile sensor movements were not affected by collisions; **(2)** Max-sum_MST with breakdowns, in which colliding agents exhibited a breakdown and stopped moving, but kept on sensing and communicating with other sensors; **(3)** DSA_MST, the standard DSA version described in [37]. **(4)** CADSA, a collision avoiding version of DSA_MST. We ranked the mobile sensors according to their indexes. Each mobile sensor updated its neighbors before moving to a new location. A mobile sensor did not move to a new location if a different mobile sensor with higher rank reported that it plans to move to the same location. **(5)** DSSA, the distributed stochastic search algorithm, designed to avoid collisions between ships [11]. DSSA allows mobile sensors to keep suggesting locations they intend to move to while checking for collisions, until they converge to a collision avoiding decision in each iteration. **(6)** Random walk, used as a baseline for the number of expected collisions.

[4]To preserve anonymity of the submission, code will be available upon publication of the paper

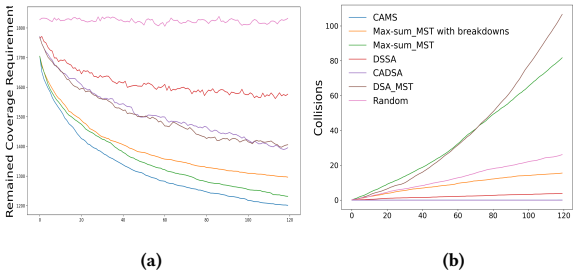


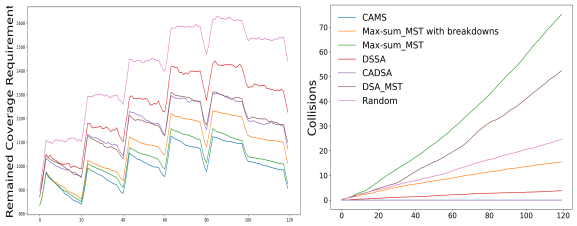
Figure 5: Remaining coverage (a), and accumulated collisions (b), as a function of the iterations, Grid-Static

Each algorithm performed 120 iterations (grid) and 40 iterations (graph), where in each iteration the mobile sensors selected locations. CAMS and Max-sum_MST performed 10 steps of Max-sum in each iteration, before the mobile sensors selected their locations. The remaining coverage in each iteration was calculated as $\sum_{T_j \in T} cr(T_j)$. We performed t-tests with $p < 0.01$ in order to evaluate statistical significance when comparing between the results produced by the different algorithms.

Figure 5(a) presents the remaining coverage requirement of the sensors performing the different algorithms as a function of the number of iterations in the *static* grid setting. It is clear that both CAMS and Max-sum_MST algorithms had a significant advantage over random walk and over all the DSA versions. Since the mobile senors in Max-sum_MST do not avoid collisions, they are less restricted and therefore the resulting coverage is significantly better than the results of the experiments that included breakdowns. Nevertheless, CAMS significantly outperformed also Max-sum_MST, which did not exhibit breakdowns, that is, CAMS was able to find collision-free solutions that their coverage results are significantly better than the solutions produced by the best algorithm that ignored collisions.

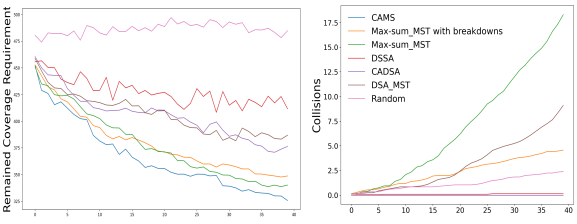
Figure 5(b) presents the number of accumulated collisions for each algorithm in these experiments, as a function of the number of iterations. Clearly, the algorithms that do not avoid collisions exhibit more collisions than the random walk. This can be explained by the attraction of mobile sensors to locations from which targets can be covered. CAMS, as well as CADSA and DSSA, do not exhibit any collisions.

Figure 6 presents the remaining coverage (a) and the number of accumulated collisions (b) of the different algorithms in the *dynamic* grid setting. Also here CAMS significantly outperforms all other algorithms in terms of remaining coverage. In addition, the gap



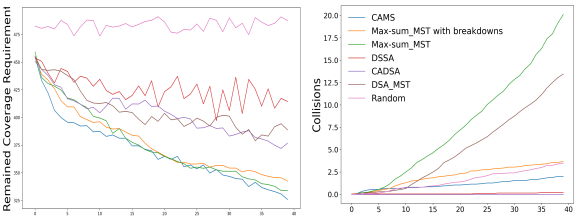
(a) (b)

Figure 6: Remaining coverage (a), and accumulated collisions (b), as a function of # iterations, Grid-Dynamic



(a) (b)

Figure 7: Remaining coverage (a), and accumulated collisions (b), as a function of # iterations, Graph-Static



(a) (b)

Figure 8: Remaining coverage (a), and accumulated collisions (b), as a function of # iterations, Grid-Dynamic

between algorithms grows with the progression of iterations, while CAMS continues to successfully adapt itself to periodic changes and maintains a consistent advantage over the other algorithms.

Figures 7 and 8 present results of experiments for random graphs in static and dynamic settings, respectively. CAMS outperforms other algorithms here as well. The number of collisions grows for the algorithms that do not prevent collisions in these experiments due to the dense environment.

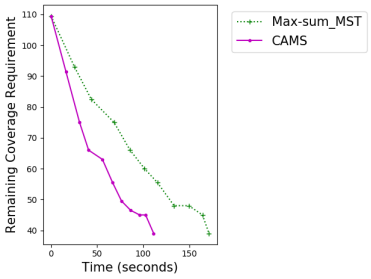


Figure 9: Remaining coverage as a function of time

6.2 Evaluation in a Physical Robotic System

In the next set of experiments our goal was to examine the practicality of CAMS in a physical multi-robot system, and specifically to evaluate the delay caused by collisions between robots in such realistic setting. Thus, we have fully implemented CAMS and Max-sum_MST on a mobile sensing team composed of three Hamster robots [4] and two targets, placed in a 4×4 grid, where the size of each cell of the grid was one square meter. The targets' ER was set to 60, and they were placed randomly in non-adjacent cells of the grid. The sensors' credibility and the number of steps the Max-sum algorithm was performed in each iteration were identical to the simulation experiments. The number of iterations performed was 10. We selected four different positions for the targets, and for each of them five different positions for the robots, resulting in 20 experiments for each algorithm.

Figure 9 presents the remaining coverage requirement as a function of the experiment execution time. Both algorithms produced the same level of coverage after completing 10 iterations. However, CAMS reached this coverage state faster.

7 CONCLUSION

An important feature of applications that include mobile sensors, is that they should avoid collisions, while optimizing coverage. CAMS achieves this challenging combination by adding to the factor-graph representation of the problem, hard constraint function-nodes, representing the locations that mobile sensors may choose to move to. In contrast to what one might expect, although this addition resulted in a much denser factor-graph including many more cycles, it did not prevent the algorithm from converging. Our theoretical analysis gave some insight to this phenomenon. We proved that in Max-sum_MST, target representing function-nodes send consistent messages throughout the algorithm run. This result explains the fast convergence of Max-sum_MST, in contrast to the behavior of standard Max-sum when solving other benchmark problems. We proved that when considering only location representing function-nodes, on tree structured graphs and on single cycle graphs the algorithm is guaranteed to converge to the optimal, collision-free solution. Moreover, adding target representing function-nodes to these graphs does not change the convergence properties. Our empirical results, revealed that the desired properties are maintained when the problem scales and in the presense of dynamic events. Most important, the advantage of Max-sum_MST over DSA_MST is maintained in the collision avoiding versions of these algorithms.

8 ACKNOWLEDGEMENTS

This research was supported in part by the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive Robotics Initiative and by the Marcus Endowment Fund both at Ben-Gurion University of the Negev, as well as BSF and ISF grands.

REFERENCES

[1] Dor Atzmon, Roni Stern, Ariel Felner, Glenn Wagner, Roman Barták, and Neng-Fa Zhou. 2020. Robust Multi-Agent Path Finding and Executing. *Journal of Artificial Intelligence Research (JAIR)* 67 (2020), 549–579.

[2] Ziyu Chen, Yanchen Deng, Tengfei Wu, and Zhongshi He. 2018. A class of iterative refined Max-sum algorithms via non-consecutive value propagation strategies. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 32, 6 (2018), 822–860.

[3] Ziyu Chen, Xingqiong Jiang, Yanchen Deng, Dingding Chen, and Zhongshi He. 2019. A Generic Approach to Accelerating Belief Propagation Based Incomplete Algorithms for DCOPs via a Branch-and-Bound Technique. In *The Thirty-Third Conference on Artificial Intelligence, (AAAI), Honolulu, Hawaii, USA*. 6038–6045.

[4] Cogniteam. 2020. *Hamster V7 Smart ROS Autonomous Ground Vehicles for Industry and Academic R & D.* https://www.hamster-robot.com/

[5] Liel Cohen, Rotem Galiki, and Roie Zivan. 2020. Governing convergence of Max-sum on DCOPs through damping and splitting. *Artif. Intell.* 279 (2020).

[6] Jonathan A DeCastro, Javier Alonso-Mora, Vasumathi Raman, Daniela Rus, and Hadas Kress-Gazit. 2018. Collision-free reactive mission and motion planning for multi-robot systems. In *Robotics research*. Springer, 459–476.

[7] Yanchen Deng and Bo An. 2020. Speeding Up Incomplete GDL-based Algorithms for Multi-agent Optimization with Dense Local Utilities. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence, (IJCAI)*. 31–38.

[8] Alessandro Farinelli, Alex Rogers, and Nick R. Jennings. 2014. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 28, 3 (2014), 337–380.

[9] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. 2008. Decentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm. In *Proceeding of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 639–646.

[10] G David Forney, Frank R Kschischang, Brian Marcus, and Selim Tuncel. 2001. Iterative decoding of tail-biting trellises and connections with symbolic dynamics. In *Codes, Systems, and Graphical Models*. Springer, 239–264.

[11] K. Hirayama, K. Miyake, T. Shiota, and T. Okimoto. 2019. DSSA+: Distributed Collision Avoidance Algorithm in an Environment where Both Course and Speed Changes are Allowed. *The International Journal on Marine Navigation and Safety of Sea Transportation* 13 (2019).

[12] Khoi D. Hoang, William Yeoh, Makoto Yokoo, and Zinovi Rabinovich. 2020. New Algorithms for Continuous Distributed Constraint Optimization Problems. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*. 502–510.

[13] Michael Hoy, Alexey S Matveev, and Andrey V Savkin. 2015. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica* 33, 3 (2015), 463–497.

[14] M. Jain, M. E. Taylor, M. Yokoo, and M. Tambe. 2009. DCOPs Meet the Real World: Exploring Unknown Reward Matrices with Applications to Mobile Sensor Networks. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, (IJCAI)*. 181–186.

[15] Md. Mosaddek Khan, Long Tran-Thanh, Sarvapali D. Ramchurn, and Nicholas R. Jennings. 2018. Speeding Up GDL-Based Message Passing Algorithms for Large-Scale DCOPs. *Comput. J.* 61, 11 (2018), 1639–1666.

[16] Y. Kim and V. R. Lesser. 2013. Improved max-sum algorithm for DCOP with n-ary constraints. In *Proceeding of the 12th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

[17] K. S. Macarthur, R. Stranders, S. D. Ramchurn, and N. R. Jennings. 2011. A Distributed Anytime Algorithm for Dynamic Task Allocation in Multi-Agent Systems. In *Proceedings of the 25th Conference of the American Association for Artificial Intelligence (AAAI)*.

[18] Duc Thien Nguyen, William Yeoh, Hoong Chuin Lau, Shlomo Zilberstein, and Chongjie Zhang. 2014. Decentralized Multi-Agent Reinforcement Learning in Average-Reward Dynamic DCOPs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 1447–1455.

[19] J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[20] Poom Pianpak, Tran Cao Son, Z O Toups, and William Yeoh. 2019. A distributed solver for multi-agent path finding problems. In *Proceedings of the First International Conference on Distributed Artificial Intelligence*. 1–7.

[21] M. Pujol-Gonzalez, J. Cerquides, P. Meseguer, J. A. Rodríguez-Aguilar, and M. Tambe. 2013. Engineering the decentralized coordination of UAVs with limited communication range. *Advances in Artificial Intelligence* (2013), 199–208.

[22] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. 2010. Decentralized Coordination in RoboCup Rescue. *Computer Journal* 53, 9 (2010), 1447–1461.

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

[24] Pierre Rust, Gauthier Picard, and Fano Ramparany. 2016. Using Message-Passing DCOP Algorithms to Solve Energy-Efficient Smart Environment Configuration Problems. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence, (IJCAI)*. 468–474.

[25] Oren Salzman and Roni Stern. 2020. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 1711–1715.

[26] Amit Sarker, Moumita Choudhury, and Md. Mosaddek Khan. 2021. A Local Search Based Approach to Solve Continuous DCOPs. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1127–1135.

[27] Álvaro Serra-Gómez, Bruno Brito, Hai Zhu, Jen Jen Chung, and Javier Alonso-Mora. 2020. With whom to communicate: learning efficient communication for multi-robot collision avoidance. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 11770–11776.

[28] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence journal (AIJ)* 219 (2015), 40–66.

[29] Roni Stern, Nathan R Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Satish Kumar, et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Twelfth Annual Symposium on Combinatorial Search*.

[30] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. 2009. Decentralised Coordination of Mobile Sensors Using the Max-Sum Algorithm. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, (IJCAI)*. 299–304.

[31] Camilla Tabasso, Venanzio Cichella, Syed Bilal Mehdi, Thiago Marinho, and Naira Hovakimyan. 2021. Time coordination and collision avoidance using leader-follower strategies in multi-vehicle missions. *Robotics* 10, 1 (2021), 34.

[32] M. E. Taylor, M. Jain, Y. Jin, M. Yokoo, and M. Tambe. 2010. When should there be a "Me" in "Team"?: distributed multi-agent optimization under uncertainty. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 109–116.

[33] Thomas Voice, Ruben Stranders, Alex Rogers, and Nicholas R. Jennings. 2010. A Hybrid Continuous Max-Sum Algorithm for Decentralised Coordination. In *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*. 61–66.

[34] Guiling Wang, Guohong Cao, Piotr Berman, and Thomas F. Laporta. 2003. A Bidding Protocol for Deploying Mobile Sensors. In *Proceedings of the 11th IEEE International Conference on Network Protocols (IEEE ICNP)*.

[35] Yair Weiss. 2000. Correctness of Local Probability Propagation in Graphical Models with Loops. *Neural Computation* 12, 1 (2000), 1–41.

[36] Harel Yedidsion, Roie Zivan, and Alessandro Farinelli. 2014. Explorative max-sum for teams of mobile sensing agents. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS*. 549–556.

[37] Harel Yedidsion, Roie Zivan, and Alessandro Farinelli. 2018. Applying max-sum to teams of mobile sensing agents. *Engineering Applications of Artificial Intelligence (EAAI)* 71 (2018), 87–99.

[38] W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. 2005. Distributed stochastic search and distributed breakout: properties, comparishon and applications to constraints optimization problems in sensor networks. *Artificial Intelligence* 161:1-2 (January 2005), 55–88.

[39] Roie Zivan, Omer Lev, and Rotem Galiki. 2020. Beyond Trees: Analysis and Convergence of Belief Propagation in Graphs with Multiple Cycles. In *Proceedings of the 34th International Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*. 7333–7340.

[40] Roie Zivan, Tomer Parash, Liel Cohen, Hilla Peled, and Steven Okamoto. 2017. Balancing exploration and exploitation in incomplete Min/Max-sum inference for distributed constraint optimization. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 31, 5 (2017), 1165–1207.

[41] Roie Zivan, Harel Yedidsion, Steven Okamoto, Robin Glinton, and Katia P. Sycara. 2015. Distributed constraint optimization for teams of mobile sensing agents. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 29, 3 (2015), 495–536.