Insights Regarding the Success of Damping in Improving Belief Propagation

Uriel Zaed
Ben-Gurion University
Beer Sheva, Israel
zaed@post.bgu.ac.il

Omer Lev
Ben-Gurion University
Beer Sheva, Israel
omerlev@bgu.ac.il

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

ABSTRACT

A common approach for solving distributed constraint optimization problems (DCOPs) is to represent them with a graphical model and to solve them with a message passing algorithm. Belief propagation is a popular and well studied such incomplete inference algorithm. Min-sum (often referred to as Max-sum) is the belief propagation version that is used for solving minimization DCOPs. Belief propagation is performed on a factor graph representation of the problem, in which the graph nodes take an active role in the algorithm, i.e., they perform calculations and exchange messages with their neighbors. Unfortunately, the standard version of Min-sum fails to converge in many cases, and produces low quality solutions. Previous studies proposed methods to encourage its convergence and improve solution quality.

Recently, empirical evidence indicated that the performance of Min-sum can be immensely improved by enhancing it with damping of beliefs (constraint costs) that are exchanged by the graph nodes. However, while this was empirically validated, a theoretical understanding of this phenomenon has not yet been established.

In this research, we present a number of theoretical and empirical results that achieve important mile-stones in understanding damping's success in improving Min-sum. These include adapting theoretical tools that were suggested for analyzing Min-sum to work with Damped Min-sum (DMS) and analyzing the effect of damping on graphs with special structures. We show that when belief propagation instantly converges, damping is redundant, and thus, the main contribution of damping is in reducing the exponential growth of the inconsistent beliefs that are propagated in the first steps of the algorithm's run.

KEYWORDS

Belief Propagation, Distributed Constraint Optimization, Min-sum, Damping

ACM Reference Format:
Uriel Zaed, Omer Lev, and Roie Zivan. 2025. Insights Regarding the Success of Damping in Improving Belief Propagation. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

Distributed Constraint Optimization Problems (DCOP) are a general model for representing and solving problems that are distributed by nature, which have a wide range of applications in artificial intelligence and multi agent systems [1, 16, 42]. Among the applications that can be represented as DCOPs are meeting scheduling, multi-agent task allocation, disaster response and operating room assignment [12, 13, 19, 24].

Two main approaches are commonly used for solving DCOPs: first, *distributed search*, in which, usually, a candidate solution is maintained and updated while traversing the solution space to find one with higher quality than those found previously [21, 23, 43, 45]. The second is *dynamic programming inference* [6, 28, 31], in which information is aggregated such that it enables the selection of high-quality solutions. These inference algorithms are rooted in techniques such as belief propagation and bucket elimination [7, 25], which are used to solve optimization problems represented by graphical models.

Probabilistic inference is the general title for problems that involve reasoning about complex distributions represented by graphical models [22]. One such problem is the maximum a posteriori (MAP) assignment problem, which seeks the most probable assignment to a set of variables [34]. It is known to be equivalent to a constraint optimization problem and the problems are easily transformed from one to the other [10, 35, 40]. Because of this close relationship, advancements in the design of inference algorithms for constraint optimization problems are expected to have broader implications for the design of algorithms for solving other combinatorial problems that can be represented by graphical models.

Min-sum (also called Max-sum when applied to maximization problems) is an incomplete inference algorithm that received considerable attention in recent years [3, 8, 10, 44]. It is designed as a message-passing algorithm in which nodes of a factor graph exchange messages with neighboring nodes, and is a version of the well-known belief propagation algorithm [25, 41], adjusted to solve constraint optimization problems (COPs) and distributed COPs (DCOPs) [2, 5, 15, 17]. It was found useful for solving multi-agent applications such as sensor systems [36, 38, 42], task allocation for rescue teams in disaster areas [30], and IoT applications [32].

Belief propagation in general (and Min-sum specifically) is known to converge to the optimal solution for problems in which the constraint graph is acyclic, but there is no such guarantee for problems which include cycles [10, 41]. Furthermore, in graphs with multiple cycles, duplicated information is propagated, leading to inaccurate and inconsistent inference [25]. Unfortunately, the underlying problem-representation graphs of many realistic applications do

include multiple cycles (e.g., [14, 23]). To improve Min-sum's performance on problems with multiple cycles, recent studies proposed methods that reduce the effect of duplicated information and trigger convergence to high quality solutions [4, 31, 47].

One such method is *damping* [20, 29, 33, 37], which has been empirically shown to overcome the double counting effect of multiple cycles when solving distributed constraint optimization problems [4, 6]. Moreover, a recent study showed that by allowing a subjective dynamic (attentive) damping factor for each edge in the factor graph, and adjusting weights that can reduce beliefs received from neighboring nodes, further improvement can be achieved [9]. However, while such attentive damping factors and belief weights were tuned using deep neural nets, showing the potential of belief propagation for solving COPs, they do not shed light on the fundamental properties that underlie this success.

In order to gain theoretical understanding of the properties of Min-sum, Zivan et al. [46] proposed the backtracking cost tree (BCT) as an analytical tool, which traces the cost accumulation procedure of the algorithm that results in beliefs sent from one node in the factor graph to another. The BCT reveals and explains some of the properties of Min-sum, such as scenarios in which it is guaranteed to converge [46] and the conditions in which belief equalities are generated in single-cycle graphs [5]. However, the structure and content of the BCT of damped Min-sum (DMS) has not yet been explored, and it is clear that to gain greater understanding of the success of DMS, one must investigate the coefficients of its BCT.

In this paper we extend the theory on belief propagation Min-sum optimization by investigating the success of damping in encouraging convergence of Min-sum to high quality solutions. Our approach is to formalize the BCT of DMS and analyze the coefficients of costs exchanged by nodes in the BCT

More specifically we:

(1) Formalize the coefficients of the components of the BCT when using DMS, as a function of the damping factor, in chain structure, single-cycle, and lemniscate graphs; and we give an overview for the general case.
(2) Prove that in a single-cycle graph, when the algorithm does not converge, the coefficients are similar to the coefficients in a specific single-cycle graph where the cycle's size is equal to a single interval of the repeated minimal route and on which the algorithm does converge.
(3) Demonstrate that on single-cycle graphs where Min-sum does not converge, the convergence achieved by DMS is a result of value equalities between the beliefs of different assignment alternatives.
(4) Prove that on problems with immediate convergence damping is not needed, regardless of the graph's structure. This result highlights the role of damping in reducing the effect of the first iterations of the algorithm in which value assignments that are not part of the optimal solution are selected and included in the BCT – reducing the overall quality of the ultimate solution.

2 BACKGROUND

A DCOP is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, C \rangle$, where $\mathcal{A}$ is a finite set of agents $\{A_1, A_2, \ldots, A_n\}$; $\mathcal{X}$ is a finite set of variables $\{X_1, X_2, \ldots, X_m\}$, where each variable is held by a single agent (an agent may hold more than one variable); $\mathcal{D}$ is a set of domains $\{D_1, D_2, \ldots, D_m\}$, where each domain D_i contains the finite set of values that can be assigned to variable X_i and we denote an assignment of value $d \in D_i$ to X_i by an ordered pair $\langle X_i, d \rangle$; and C is a set of constraints (relations), where each constraint $C_j \in C$ defines a non-negative *cost* for every possible value combination of a set of variables and is of the form $C_j : D_{j_1} \times D_{j_2} \times \ldots \times D_{j_k} \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary constraint* refers to exactly two variables and is of the form $C_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{0\}$. In our discussion of Min-sum we often refer to C_{ij} as the *cost table* and to $C_{ij}[w, r]$ as the entry corresponding to $w \in D_i$ and $r \in D_j$.

A *binary DCOP* is a DCOP in which all constraints are binary. Agents are *neighbors* if they are involved in the same constraint. A *partial assignment* (PA) is a set of value assignments to variables, in which each variable appears at most once. A constraint $C_j \in C$ of the form $C_j : D_{j_1} \times D_{j_2} \times \ldots \times D_{j_k} \rightarrow \mathbb{R}^+ \cup \{0\}$ is *applicable* to PA if each of the variables $X_{j_1}, \ldots, X_{j_k}$ are included in the PA. The *cost of a partial assignment* PA is the sum of all applicable constraints to PA over the value assignments in PA. A *complete assignment* (i.e., *solution*) is a partial assignment that includes all variables ($\mathcal{X}$). An *optimal solution* is a complete assignment with minimal cost.

For simplicity, we assume that each agent holds exactly one variable (i.e., $n = m$) and we focus on binary DCOPs. These assumptions are common in DCOP literature (e.g., [27, 43]).

2.1 Min-sum Belief Propagation

Min-sum operates on a *factor-graph*, a bipartite graph in which the nodes represent variables and constraints [18]. Each variable-node, representing a variable of the original DCOP, is connected to all function-nodes representing constraints that it is involved in. Similarly, a function-node is connected to all variable-nodes involved in the constraint it represents. Variable-nodes and function-nodes take an active role in Min-sum, i.e., they can send and receive messages, and can perform computation. When used to solve a DCOP, each node's role is performed by an autonomous agent.

In Min-sum, 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 (or the belief of a cost) for each value in D_X. In the first iteration, all nodes assume that all messages they previously received (in iteration 0) include vectors of zeros. A message $Q^i_{X \rightarrow F}$ that variable-node X sends to function-node F in iteration i is formalized as follows: $Q^i_{X \rightarrow F} = \sum_{F' \in F_X \setminus \{F\}} R^{i-1}_{F' \rightarrow X} - \alpha$, where F_X is the set of function-node neighbors of variable-node X and $R^{i-1}_{F' \rightarrow X}$ is the message sent to variable-node X by function-node F' in iteration $i-1$. α is a constant that is reduced from all costs included in the message in order to prevent the costs from growing arbitrarily large.

A message $R^i_{F \rightarrow X}$ sent from a function-node F to a variable-node X in iteration i, includes for each value $x \in D_X$: $R^i_{F \rightarrow X} = \min_{PA_{-X}} cost(\langle X, x \rangle, PA_{-X})$, where PA_{-X} is a possible combination of value assignments to variables involved in F not including X. $cost(\langle X, x \rangle, PA_{-X})$ represents the cost of a partial assignment $a = \{\langle X, x \rangle, PA_{-X}\}$, which is: $f(a) + \sum_{X' \in X_F \setminus \{X\}, \langle X', x' \rangle \in a} (Q^{i-1}_{X' \rightarrow F})[x']$, 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 $(Q^{i-1}_{X'\to F})[x']$ is the cost that was received in the message sent from variable-node X' in iteration $i-1$, for the value x' that is assigned to X' in a. X selects its value assignment $\hat{x} \in D_X$ following iteration k as follows: $\hat{x} = \underset{x\in D_X}{\operatorname{argmin}} \sum_{F\in F_X} (R^k_{F\to X})[x]$.

2.2 Single-cycle factor graphs

Belief propagation is known to converge to the optimal solution in linear time when solving factor graphs with a tree structure (includes no cycles). For factor graphs with a single-cycle, if belief propagation converges at all, it converges to the optimal solution [11, 39] (though it is not guaranteed to converge on such factor graphs). When it does not converge it periodically changes its selected assignments.

To explain this behavior, Forney et al. [11] show the similarity of the performance of the algorithm on a cycle to its performance on a chain-structured graph, with nodes similar to the nodes in the cycle, but the chain length is the number of iterations performed by the algorithm so far. Consider a sequence of messages starting at the first node of the chain and heading towards the other end. Each message conveys beliefs accumulated from the costs added by function-nodes. Specifically, each function-node adds a cost to each belief, which is the constraint cost of a pair of values assigned to its neighboring variable-nodes. Each such sequence of cost accumulations (i.e., each route) must become periodic at some point, and the minimal belief is generated by the minimal periodic route. If this periodic route is consistent, i.e., if the set of assignments implied by the costs within it contain the same value for each variable, the algorithm converges and the implied assignment is the optimal solution; otherwise, it does not converge [11].

2.3 Damped Min-sum (DMS)

In order to add damping to Min-sum, a parameter $\lambda \in [0, 1)$ is used. Before sending a message in iteration k, a node performs calculations as in standard Min-sum. We use $\widehat{m^k_{i\to j}}$ to denote the result of the calculation made in standard Min-sum for the content of a message intended to be sent from node i to node j in iteration k and $m^{k-1}_{i\to j}$ to denote the message sent by i to j at iteration $k-1$. The message sent by i to j at iteration k is calculated as follows: $m^k_{i\to j} = \lambda m^{k-1}_{i\to j} + (1-\lambda)\widehat{m^k_{i\to j}}$. Thus, λ expresses the weight given to previously performed calculations with respect to the most recent calculation performed. When $\lambda = 0$ the resulting algorithm is standard Min-sum.

2.4 Backtrack Cost Trees

For analyzing the behavior of Min-sum, the use of a *backtrack cost tree* (BCT) was proposed by Zivan et al. [46]. It allows tracing for each belief the entries in the cost tables held by function-nodes that were used to compose it.

A BCT is defined for a belief that appears in a message (either from some variable X_i to a function-node F_{ij}, or from some function-node F_{ij} to a variable X_i). The belief is on the cost of assigning some value $x \in D_i$ to variable X_i. Without loss of generality, we will elaborate on messages from variables to function-nodes.

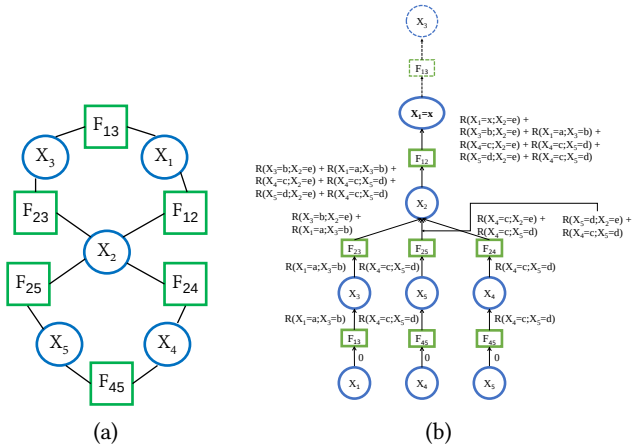

Figure 1: (a) A lemniscate factor-graph. (b) An example of a BCT for a belief in the message sent from X_1 to the function-node F_{13} at time $t = 6$ in the lemniscate depicted on the left.

The belief is a sum of various components from which the BCT is composed. At the root is the cost for a decision to assign some value to a variable at a time t and the directed edges from its children in the tree include the beliefs that were summed to generate that belief. These edges lead to nodes representing the neighbors from which the parent node received messages in time $t-1$. Each of those nodes is connected to the nodes from which it received messages at time $t-2$, with the edges containing the beliefs that were passed to it. The tree leaves represent the nodes at time 0 (see Figure 1(b)).

For a single-cycle factor graph, the BCT for every belief is a chain. Factor graphs with multiple cycles include variable-nodes with more than two neighbors, and thus, the BCTs of their beliefs include nodes with multiple children.

For each BCT, there is an implied assignment tree including the value assignments that the variables at each time-point of the tree would need to be assigned in order to incur the costs included in the BCT. The value assignment selected by a variable at time t is the one with the minimal sum of beliefs sent to the corresponding variable-node at iteration $t-1$.

3 FORMALIZING THE COEFFICIENTS OF A DMS BCT

The difference between regular Min-sum BCT and DMS BCT is the weight that every node in the BCT is given in the calculation of the belief at the root of the BCT. In standard Min-sum BCT, the belief is a sum of the costs in all nodes of the BCT, i.e., the cost taken from the entries in the function-node cost table and added in the node of the BCT, is just itself. Thus, the belief is composed of the sum of cost table entries of function-nodes, each multiplied by the number of nodes in the BCT in which this cost table entry was added. In contrast, in a BCT that is used to analyze the performance of DMS, the cost added in each such BCT node is multiplied by a coefficient that is a result of damping. These coefficients are calculated using a non trivial formula (as past messages are repeatedly added, multiplied by an updated coefficient). Thus, to understand

the difference between the BCTs, we must formalize the coefficients within the BCT generated by DMS. We begin by examining graphs with simple structures, before elaborating on the general case.

3.1 DMS Coefficients for Chain-Structured Graphs

Consider a factor graph with n variable-nodes, structured as a chain. W.l.o.g., we assume an order on the variable-nodes coinciding with their indexes. While messages are sent in both directions, in a chain structure factor graph a message does not affect the messages in the opposite direction [5], thus, we will only consider the messages in ascending index order. Denote by $R^k_{ij\to j}$, $1 \le i < n$, $j = i+1$ the message sent by function-node F_{ij} to X_j in iteration k and $v^k_{ij\to j}$, the vector it is carrying (to avoid redundancy, we will use $R^k_{ij\to j}$ and $v^k_{ij\to j}$ for the message and the vector it carries interchangeably). Recall that we use $C_{ij}[w,r]$ to denote the entry in the cost table of F_{ij} corresponding to values $w \in D_i$ and $r \in D_j$ and similarly, that $Q^k_{i\to ij}[w]$ and $R^k_{ij\to j}[r]$ correspond to the relevant entries in the vectors included in the messages sent in iteration k. Thus:

$$R^k_{12\to 2}[r] = (1-\lambda)\sum_{q=1}^{k}\lambda^{q-1}\min_{w\in D_1} C_{12}[w,r]$$

This is because $R_{12\to 2}$ depends solely on the cost table C_{12}. In each iteration, we compute a new message as a weighted sum of all previous computations, where $(1-\lambda)$ is the weight for the current computation and λ is the weight of the message from the previous iteration. The min operation within the sum corresponds to the standard Min-Sum message computation, which selects the minimal value for the sending variable.

For F_{ij} ($i \ge 2$), we must also account for the incoming message $Q_{i\to ij}$ from the previous variable node X_i and the previous message of F_{ij}, in iteration $k-1$.

$$R^k_{ij\to j}[r] = (1-\lambda)\sum_{q=1}^{k}\lambda^{q-1}\min_{w\in D_1}\left(C_{ij}[w,r] + Q^{k-1}_{i\to ij}[w]\right)$$

Next, we consider the limit of the iterations of each message R:

$$\lim_{k\to\infty} R^k_{12\to 2}[r] = \min_{w\in D_1} C_{12}[w,r]$$
$$\lim_{k\to\infty} R^k_{ij\to j}[r] = \min_{w\in D_1}\left(C_{ij}[w,r] + Q^{k-1}_{i\to ij}[w]\right)$$

We can observe that the use of damping on a chain results in each message converging to the same value as it would without damping. This indicates that applying damping in chain structures does not significantly alter the behavior of the algorithm.

Furthermore, in trees structures, while the function nodes receive additional vectors their behavior essentially remains unchanged, maintaining the same overall message dynamics.

3.2 DMS Coefficients for Single-Cycle Graphs

When Min-sum solves a single-cycle factor graph, it reaches a repeated minimal route that it executes until termination [5, 11, 39]. Convergence is achieved if this route is consistent, i.e., each variable-node is assigned the same value. Otherwise, it repeats an inconsistent minimal route, whose length is the number of values assigned to a variable-node in the route, multiplied by the size of the cycle [5]. Before reaching the infinitely repeated minimal route, the algorithm may traverse value assignments that are not part of the repeated minimal route, which we term as the *tail* [5].

3.2.1 DMS Coefficients for Single-Cycle Structured Graph with a Consistent Minimal Route. For simplicity, we begin by analyzing the case in which the algorithm converges instantly, i.e., there is no tail. W.l.o.g., we will consider a single-cycle factor graph with three variable-node, so the algorithm converges right away to the optimal solution, which we term $X_1 = 0, X_2 = 0, X_3 = 0$.

Our approach is to investigate the coefficients of the costs added in the BCT, separately, for each function-node. W.l.o.g., we will analyze the generation of the coefficient of the entry $C_{12}[0,0]$ and we denote the cost in this entry by $c_{[0,0]}$.

Iteration 1: $(1-\lambda)c_{[0,0]}$
Iteration 2: $\lambda(1-\lambda)c_{[0,0]} + (1-\lambda)c_{[0,0]} = c_{[0,0]}(1-\lambda)(1+\lambda)$
Iteration 3: $\lambda(1-\lambda)(1+\lambda)c_{[0,0]} + (1-\lambda)c_{[0,0]} = c_{[0,0]}(1-\lambda)(1+\lambda+\lambda^2)$
Iteration 4: $\lambda c_{[0,0]}(1-\lambda)(1+\lambda^2) + (1-\lambda)c_{[0,0]} = c_{[0,0]}(1-\lambda)(1+\lambda+\lambda^2+\lambda^3)$
...
Iteration 7: $\lambda c_{[0,0]}(1-\lambda)(1+\lambda^2+\lambda^3+\lambda^4+\lambda^5) + (1-\lambda)(c_{[0,0]} + (1-\lambda)^3 c_{[0,0]}) = c_{[0,0]}(1-\lambda)(1+\lambda+\lambda^2+\lambda^3+\ldots+\lambda^6+(1-\lambda)^3)$
...
Iteration 13: $\lambda c_{[0,0]}(1-\lambda)(1+\lambda^2+\ldots+\lambda^{11}+(1-\lambda)^3(1+4\lambda+10\lambda^2+20\lambda^3+35\lambda^4+56\lambda^5)) + 1-\lambda)(c_{[0,0]} + c_{[0,0]}(1-\lambda)^3(1+3\lambda+6\lambda^2+10\lambda^3+15\lambda^4+21\lambda^5+28\lambda^6+(1-\lambda)^6) = c_{[0,0]}(1-\lambda)(1+\lambda+\lambda^2+\lambda^3+\ldots+\lambda^{12}+(1-\lambda)^3(1+4\lambda+10\lambda^2+20\lambda^3+35\lambda^4+56\lambda^5+84\lambda^6)+(1-\lambda)^6)$
...

We focus on iterations 7 and 13 because the cycle size is 6, thus, these iterations follow the first and second times that the cycle has been completed by the algorithm, respectively. This analysis leads to the following general coefficient Γ_1 formula in a cycle including n variable-nodes at iteration k:

$$\Gamma_1 = (1-\lambda)\cdot\sum_{q=1}^{k}\lambda^{q-1} + (1-\lambda)^{n+1}\sum_{q=1}^{k-2n}\binom{q+n-1}{n}\lambda^{q-1} + (1-\lambda)^{2n+1}\sum_{q=1}^{k-4n}\binom{q+2n-1}{2n}\lambda^{q-1}\ldots$$

The structure of the equation is based on the iterative logic of the algorithm. Each time a route returns to its starting point, a new term is added to the equation, marking the completion of a cycle. The coefficient for each newly added term is determined by the expression $(1-\lambda)$, raised to a power that corresponds to the number of function nodes traversed, and is also influenced by the expression $(1+\lambda\cdot c_1+\lambda^2\cdot c_2+\lambda^3\cdot c_3+\ldots)$. As the process iterates, past messages are progressively combined with the current value, causing the power of λ to increase. The numerical constants are derived from the binomial coefficient $\binom{q+\lfloor\frac{k-1}{2n}\rfloor n-1}{\lfloor\frac{k-1}{2n}\rfloor n}$, which is influenced by the number of function nodes traversed in the process.

To generalize this process, each part, representing a set of messages during one cycle, is:

$$(1-\lambda)^{\lfloor\frac{k-1}{2n}\rfloor n+1}\sum_{q=1}^{k-\lfloor\frac{k-1}{2n}\rfloor 2n}\binom{q+\lfloor\frac{k-1}{2n}\rfloor n-1}{\lfloor\frac{k-1}{2n}\rfloor n}\lambda^{q-1}$$

To perform a more in-depth analysis of the coefficient, we analyzed the changes in coefficient values across iterations with different damping factors. Figure 2 demonstrates that these coefficients

maintain a monotonically increasing, approximately linear trend, regardless of the damping factor used. Thus, the main claim used in Forney et al. [11], Weiss [39] to prove the optimally of the solution of belief propagation when it converges on single-cycle graphs is maintained in the presence of damping.

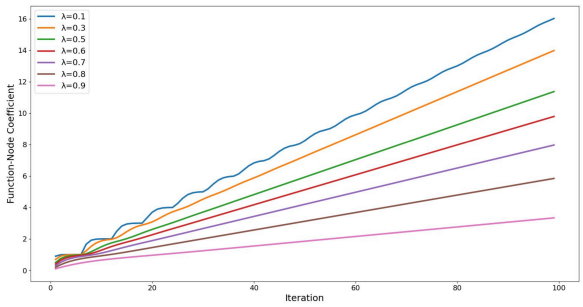

Figure 2: Function-Node coefficient

In order to understand the relationship between this coefficient and the coefficient of each cost added in the BCT nodes from which it is composed, we formalize its components, each representing the factor that a particular function-node cost table entry is multiplied by in the different levels of the BCT. Thus, we denote the coefficients of the three function-nodes α, β and γ, and w.l.o.g. we investigate the componnents that are composed to generate α. These are denoted by: $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_m$ where α_1 represents the contribution from (or the weight of) the first function-node cost table entry added in the BCT node that is the farthest from the root (i.e., closest to the leaf – the "oldest" one) in the BCT, α_2 from the one above it that refers to the same entry, and so forth until α_m, which is the closest to the root.

We formalize α_1 at iteration k as follows:

$$\alpha_i^k = (1-\lambda)\begin{cases} \sum_{q=1}^{k} \lambda^{q-1} & , \lfloor \frac{k-1}{2n} \rfloor = 0 \\ \sum_{q=k-2n}^{k} \lambda^{q-1} & , \text{otherwise} \end{cases}$$

$$+(1-\lambda)^{n+1}\begin{cases} 0 & , \lfloor \frac{k-1}{2n} \rfloor < 1 \\ \sum_{q=1}^{k-2n} \binom{q+n-1}{n} \lambda^{q-1} & , \lfloor \frac{k-1}{2n} \rfloor = 1 \\ \sum_{q=k-4n}^{k-2n} \binom{q+n-1}{n} \lambda^{q-1} & , \text{otherwise} \end{cases}$$

$$+(1-\lambda)^{2n+1}\begin{cases} 0 & , \lfloor \frac{k-1}{2n} \rfloor < 2 \\ \sum_{q=1}^{k-4n} \binom{q+2n-1}{2n} \lambda^{q-1} & , \lfloor \frac{k-1}{2n} \rfloor = 2 \\ \sum_{q=k-6n}^{k-4n} \binom{q+2n-1}{2n} \lambda^{q-1} & , \text{otherwise} \end{cases}$$

$$+\{ \ \ldots$$

Since we are dealing with a single-cycle factor graph with a consistent minimal route, the BCT exhibits a repeating pattern across the cycle's length. In other words, the expression referring to the first $2n$ iterations will always be the same in α_m. If m is large enough, then the expression representing the iterations $2n+1$ to $4n$ will always be the same in α_{m-1} and so forth. In other words, for every component α_1 to α_m in every iteration the value added to α_i, $2 \le i \le m$ is reduced from α_{i-1}. in other words $\alpha_i^k = \alpha_{i-1}^{k-2n}$ for any $k > 2n$.

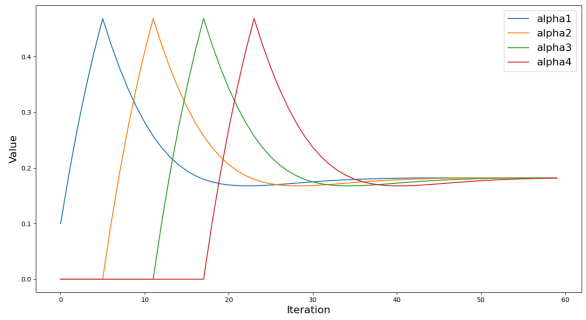

Figure 3: $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ as a function of the number of iterations, on a cycle with three function-nodes, with a damping factor $\lambda = 0.9$.

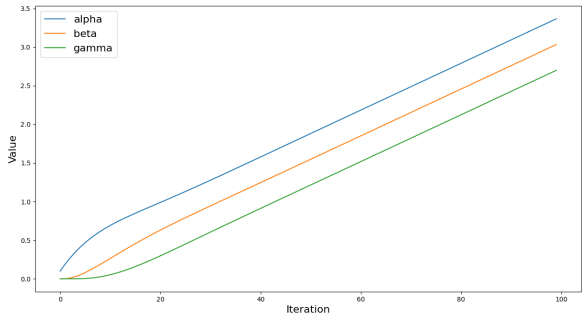

Figure 4: α, β, γ as a function of the number of iterations, on a cycle with three function-nodes, with a damping factor $\lambda = 0.9$.

Thus, for the coefficient $\alpha^k = \sum_{i=1}^{m} \alpha_i^k$, $m = \lfloor \frac{k-1}{2n} \rfloor + 1$, the difference between its value in iteration k and in iteration $k-1$, i.e., $\alpha^k - \alpha^{k-1}$ equals the last expression of α_1, that represents the values added to it in the k modulo $2n$ iterations. Formally, this expression is calculated as follows:

$$(1-\lambda)^{\lfloor \frac{k-1}{2n} \rfloor n+1} \sum_{q=1}^{k-\lfloor \frac{k-1}{2n} \rfloor 2n} \binom{q+\lfloor \frac{k-1}{2n} \rfloor n-1}{\lfloor \frac{k-1}{2n} \rfloor n} \lambda^{q-1}$$

Our empirical results, shown in Figure 3, suggest that α_1 converges to a constant value. Hence, the rest α_i values eventually converge to the same constant. As a result, we can deduce that the coefficient exhibits linear behavior, as demonstrated in Figure 4, which presents the three coefficients α, β and γ as a function of the number of iterations, when the algorithm solves a three function-node cycle graph.

Our most important empirical result is presented in Figure 5. It presents the results of a sample of many experiments we performed on cycles with different sizes and different convergence properties, all reporting similar results. Figure 5 shows the sum of beliefs for each variable nodes, as a function of the number of iterations of the algorithm, when solving a three function-node cycle graph. The first, (a), is a graph on which the algorithm converges right away to a consistent route, i.e. to the optimal solution. The second, (b), is a cycle with a consistent route, which does, however, includes a tail (that is, the algorithm does not converge right away). Similar results

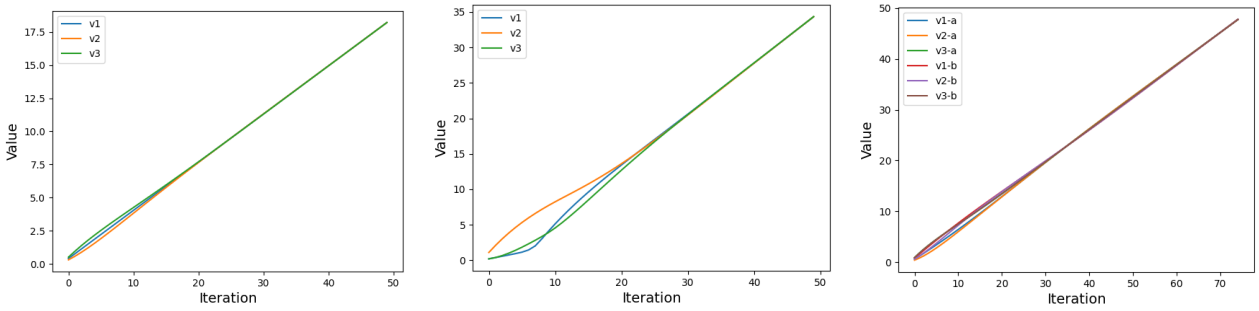

Figure 5: Sum of beliefs as a function of the number of iterations for (a) three value assignments of a three function-node cycle with a consistent minimal route and immediate convergence; (b) three value assignments of a three function-node graph with a consistent minimal route, but with a tail (not immediate convergence); (c) Variable assignment of six values in a three function-node cycle graph – an inconsistent minimal route – and with no tail.

were obtained for all single-cycle graphs that include a consistent minimal route, regardless of their size. Intuitively, this happens because for each variable node, all cost table entries summed to calculate the beliefs are damped an equal number of times, which is the size of the cycle. For example, consider the three cycle factor graph on the top of Figure 6. It includes three variable nodes and three function nodes. If you consider the distance of the function nodes from X_1 you have F_{12} and F_{13}, which adjacent to X_1, therefore their distance from X_1 on one side includes one damping operation, but on the other side, five damping operations. F_{23} on the other hand is on the other side of the cycle and every cost sent from F_{23} to X_1 goes through three damping operations, regardless of the direction. Thus, all costs are damped 6 times all together on both directions. Moreover, this seems to indicate that we see *assignment equality*, i.e., variables' costs are the same, and the choice of variable is decided based on tie-breaking.

3.2.2 DMS Coefficients of Inconsistent Routes for Single-Cycle Graphs. As previously noted, in single-cycle graphs, Min-sum can encounter an inconsistent minimal route, which oscillates between multiple entries at each function node, and as a result, different values are selected to be assigned to variables in different iterations. Cohen et al. proved that two entries of the same cost table of a function-node, which are both included in such an inconsistent minimal route, cannot be in the same row or column of the cost table [5].

We proved above that when DMS is applied to graphs where Min-sum converges, it converges such that the sum of beliefs received by each variable node for the value assignment in the optimal solution is equal to the sum of beliefs that the other variable-nodes receive for their value assignments in the optimal solution (without DMS).

Next, we will prove the same phenomenon occurs when the algorithm oscillates for all values that are included in the minimal route, i.e., the algorithm converges to a state in which the sum of beliefs is equal for all values that are included in the minimal route, whether it is consistent or not. Thus, in cases where it is not consistent, and more than one value is included in the minimal route in each domain, damping apparently results in assignment equality (since our empirical simulations indicate for every variable-node there are multiple values in its domain that the agents receives for them the same lowest sum of beliefs).

Lemma 3.1. *For every single-cycle graph G, on which the repeated minimal route is inconsistent, there is a single-cycle graph G' in which the same minimal route is consistent. Moreover, the number of function-nodes in graph G' is the number of cost table entries in a single interval of the minimal route in graph G.*

Proof. Consider a single-cycle graph G with n function-nodes on which Min-sum converges to an inconsistent minimal route, oscillating among x entries at each function node. We generate a new single-cycle graph G' with $n \cdot x$ function nodes. Let the original function nodes be $F_{12}, F_{23}, ..., F_{n1}$. Thus, the value assignments induced by the minimal route include $V_{1_a}, V_{2_a}, ..., V_{n_a}, V_{1_b}, V_{2_b}, ..., V_{n_b}, ..., V_{1_x}, V_{2_x}, ..., V_{n_x}$. We create G' by generating a variable-node for each of the value assignments induced by the minimal route. Then, in each function-node connecting two consecutive variable-nodes, we change the cost of every entry that is not included in the minimal route to the maximal cost in the original cost table plus one. If the minimal route included a tail, the entries of the tail are not changed as well. Obviously, the minimal route in G is a minimal route in G', and moreover, no other route in G' can have a smaller normalized cost than the minimal route in G. □

Figure 5 (c) demonstrates how the sum of beliefs for all value assignments induced by the inconsistent minimal route converge to the same cost.

Theorem 3.2. *The beliefs generated by DMS on a single-cycle graph G with an inconsistent minimal route, corresponding to the value assignments included in the minimal inconsistent route, are identical to the beliefs corresponding to the minimal route, sent by DMS on a single-cycle graph G' in which the same minimal route is consistent.*

Proof. By construction (as described in the proof of Lemma 3.1) the algorithm follows the same minimal route, the same entries in the function-node cost-tables are accumulated and the beliefs are multiplied by the same damping factors. □

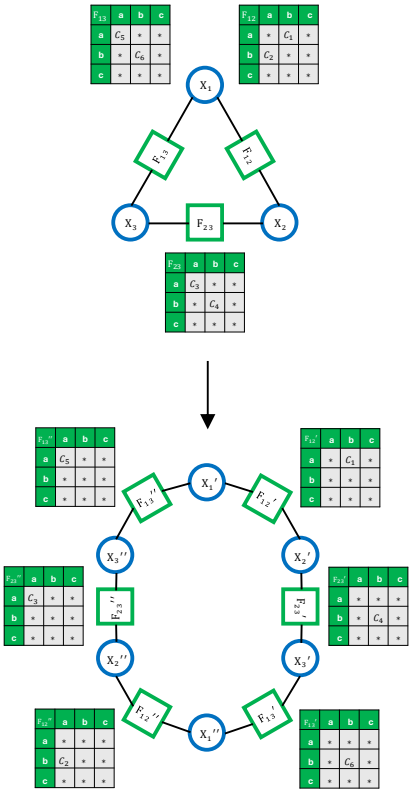

Figure 6: Inconsistent route in a single-cycle graph example

As an example, examine Figure 6, which includes a single-cycle graph, composed of 3 function-nodes and three variable nodes. In this case, the algorithm converges to an inconsistent route, denoted as, $c_1 \rightarrow c_4 \rightarrow c_6 \rightarrow c_2 \rightarrow c_3 \rightarrow c_5 \rightarrow c_1$.... We construct a single-cycle graph with 6 function-nodes as described in Lemma 3.1. This single-cycle factor graph cycle includes a consistent route that is identical to the minimal inconsistent route in the original factor graph. Consequently, the algorithm converges when solving the six variable-node factor graph to a consistent route that accurately reflects the dynamics of the initial inconsistent route.

3.3 Formalizing the Coefficients of a DMS BCT for Multiple-Cycles Graphs

As noted above, empirical evidence in previous studies revealed that damping mitigates the duplicated information phenomenon when belief propagation is applied to graphs with multiple cycles [26]. Thus, DMS outperforms Min-sum when solving problems that their underlying representing factor graph includes multiple cycles [4], but it is not clear how this is achieved. To extend our understanding of this phenomenon, we examine a simple case: a symmetric lemniscate factor graph (∞-shaped graph).

As with single-cycle graphs, examine the case in which the algorithm converges instantly to the optimal solution (i.e., no tail). Consider a leminscate factor graph with three variable-nodes in each cycle (Figure 1). We again investigate the coefficients of the costs added in the BCT. Let us examine the generation of the coefficient of the entry $C_{13}[0,0]$ in the message $R^k_{13\rightarrow 3}[r]$. Our analysis leads to the following general coefficient Ψ_1 formula at iteration k:

$$\begin{aligned}\Psi_1 = (1-\lambda)c_{[0,0]}\Big(&\sum_{i=1}^{k}\lambda^{i-1}\\ &+(1-\lambda)^n\sum_{i=1}^{k-2n}\binom{i+n-1}{n}\lambda^{i-1}\\ &+5\cdot(1-\lambda)^{2n}\sum_{i=1}^{k-4n}\binom{i+2n-1}{2n}\lambda^{i-1}\\ &+13\cdot(1-\lambda)^{3n}\sum_{i=1}^{k-6n}\binom{i+3n-1}{3n}\lambda^{i-1}\\ &+41\cdot(1-\lambda)^{4n}\sum_{i=1}^{k-8n}\binom{i+4n-1}{4n}\lambda^{i-1}\ldots\Big)\end{aligned}$$

The only difference between the coefficients in a lemniscate and those shown above for single-cycle graphs are the constants that multiply each part of the formula (in this example: 1, 1, 5, 13, 41,...). To gain deeper insights into the series of constants, a detailed exploration of the BCT is essential (Figure 7). For each iteration of the BCT, we identify and enumerate the relevant nodes that contribute to our coefficients, which are marked in red. These nodes define each constant in the series. Through a systematic approach utilizing iterative bounded Depth-First Search (DFS), we can accurately compute this series of constants.

This method allows us to compute the parameters of general graph structures, offering a versatile approach without limiting ourselves to a specific graph structure. This provides a solid foundation for analyzing complex structures and contributes to a broader understanding of the underlying patterns in the BCT in the future.

4 DMS CONVERGENCE

Beyond examining the convergence of DMS using simulations, we are able to present a more general statement.

Theorem 4.1. *Min-sum algorithm will converge to the optimal solution when there is no tail (i.e., when convergence to the final route is immediate). This result holds irrespective of the damping factor used or the structure of the graph.*

Proof. Using induction on k, the number of iterations of the BCT (that is, its depth):

Base Case: Consider a BCT after 1 iteration, meaning a single function node. In this trivial case, the algorithm will calculate the message by the minimal entries. And as there is no tail, the minimal belief assignment is a part of the optimal solution. This holds true regardless of the damping factor, as all messages are multiplied by the same constant $(1-\lambda)$.

Induction assumption: Assume that for a BCT after k iterations, Min-sum created beliefs, and the minimal belief assignment of the root is a part of the optimal solution, regardless of the damping factor or the structure of the graph.

Induction Step: Consider a BCT after $k+1$ iteration. We need to show that Min-sum algorithm created the minimal belief assignment of the root that is a part of the optimal solution, regardless of the damping factor or the structure of the graph. By adding one iteration to the BCT, the leaves are function nodes. The message each function node creates contains a belief that is part of the minimal belief assignment. Because there is no tail, we have only one minimal route that creates the minimal belief assignment of the

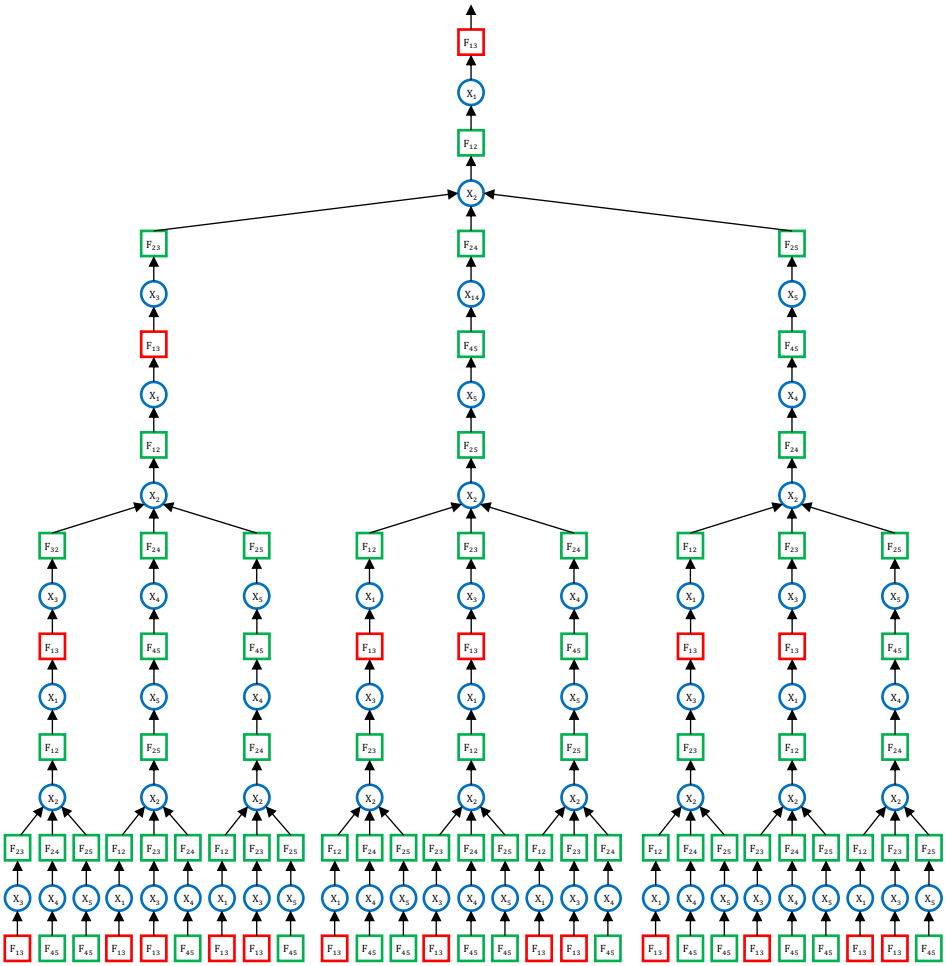

Figure 7: BCT of a leminscate factor graph with three function nodes in each cycle

root. Therefore, the beliefs that the leaves send are in the same route of the minimal route (if they were not, the algorithm would have a tail). Additionally, damping only multiplies the leaf function node messages by the same constant, meaning that we still maintain the minimal belief assignment as part of the optimal solution. Following the leave layer, in the rest of the BCT the minimal route is the optimal solution according to the induction's assumption. We note that there isn't any assumption on the graph structure. □

5 CONCLUSION

Belief propagation is a well-known and widely used algorithm for solving combinatorial optimization problems that can be represented by graphical models. While the theoretical knowledge regarding this algorithm is limited, empirical evidence indicate that the use of damping much improves its outcome.

In this paper, we presented theoretical and empirical results that extend the knowledge regarding the reasons for the success of damping to improve belief propagation. First were able to detail formulas that calculate the coefficients of the costs which are accumulated by the algorithm, and their multiplication as a result of damping. Then, we demonstrate that when damping is used and the algorithm solves a single-cycle graph, the beliefs for all values that are included in the minimal repeated route converge to the same value. Thus, when the minimal route is inconsistent, these empirical results indicate that it converges to assignment equality (agents cannot tell which of the values that belong to the minimal route to assign to their variables). Finally, we prove – for all graph topologies – that when the graph does not include a tail, i.e., the algorithm converges right away to the optimal solution, damping is not required. Thus, we conclude that the key role of damping in improving belief propagation is in eliminating the effect of the initial inconsistent part of the route, i.e., the tail. By understanding which in which cases damping is *not* crucial to an optimal outcome, we hope to help future research to particular issues which we now understand to be key to its success.

REFERENCES

[1] Kyle E. C. Booth and J. Christopher Beck. 2019. A Constraint Programming Approach to Electric Vehicle Routing with Time Windows. In *Proceedings of the 16th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*. 129–145.

[2] I. Brito and P. Meseguer. 2010. Improving DPOP with function filtering. In *9th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*. 141–148.

[3] Jesús Cerquides, Rémi Emonet, Gauthier Picard, and Juan A. Rodríguez-Aguilar. 2018. DECIMAXSUM: Using Decimation to Improve Max-Sum on Cyclic DCOPs. In *Artificial Intelligence Research and Development - Current Challenges, New Trends and Applications, in Proceedings of the 21st International Conference of the Catalan Association for Artificial Intelligence, CCIA 2018*. 27–36.

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

[5] Erel Cohen, Omer Lev, and Roie Zivan. 2023. Separate but Equal: Equality in Belief Propagation for Single Cycle Graphs. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*. 3924–3931.

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

[7] Rina Dechter. 1999. Bucket Elimination: A Unifying Framework for Reasoning. *Artificial Intelligence* 113, 1-2 (1999), 41–85.

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

[9] Yanchen Deng, Shufeng Kong, Caihua Liu, and Bo An. 2022. Deep Attentive Belief Propagation: Integrating Reasoning and Learning for Solving Constraint Optimization Problems. In *Proceedings of the Thirty-Sixth Conference on Neural Information Processing Systems (NeurIPS)*.

[10] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas 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.

[11] 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*. 239–264.

[12] Noam Gaon, Yuval Gabai Schlosberg, and Roie Zivan. 2023. Scheduling operations in a large hospital by multiple agents. *Engineering Applications of Artificial Intelligence (EAAI)* 126(D) (2023).

[13] I.P. Gent and T. Walsh. 1999. *CSPLib: a benchmark library for constraints*. Technical Report. Technical report APES-09-1999. Available from http://csplib.cs.strath.ac.uk/.

[14] Amir Gershman, Amnon Meisels, and Roie Zivan. 2009. Asynchronous Forward Bounding. *Journal of Artificial Intelligence Research (JAIR)* 34 (2009), 25–46.

[15] Supriyo Ghosh, Akshat Kumar, and Pradeep Varakantham. 2015. Probabilistic Inference Based Message-Passing for Resource Constrained DCOPs. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, (IJCAI)*. 411–417.

[16] Arnoosh Golestanian, Giovanni Lo Bianco, Chengyu Tao, and J. Christopher Beck. 2023. Optimization Models for Pickup-And-Delivery Problems with Reconfigurable Capacities. In *Proceedings of the 29th International Conference on Principles and Practice of Constraint Programming (CP)*. 17:1–17:17.

[17] Md. Mosaddek Khan, Long Tran-Thanh, William Yeoh, and Nicholas R. Jennings. 2018. A Near-Optimal Node-to-Agent Mapping Heuristic for GDL-Based DCOP Algorithms in Multi-Agent Systems. In *Proceedings of the 16th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*. 1604–1612.

[18] Frank R. Kschischang, Brendan J. Frey, and Hans A. Loeliger. 2001. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory* 47:2 (2001), 181–208.

[19] Maya Lavie, Tehila Caspi, Omer Lev, and Roie Zivan. 2023. Ask and You Shall be Served: Representing & Solving Multi-agentOptimization Problems with Service Requesters and Providers. In *Proceedings of the 22'nd International Conference on Autonomous Agents and Multiagent Systems, (AAMAS)*. 77–85.

[20] Nevena Lazic, Brendan J. Frey, and Parham Aarabi. 2010. Solving the uncapacitated facility location problem using message passing algorithms. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 429–436.

[21] Jimmy Ho-Man Lee, Pedro Meseguer, and Wen Su. 2015. Adding laziness in BnB-ADOPT+. *Constraints* 20, 2 (2015), 274–282.

[22] Radu Marinescu and Rina Dechter. 2009. AND/OR Branch-and-Bound search for combinatorial optimization in graphical models. *Artificial Intelligence* 173, 16-17 (2009), 1457–1491.

[23] Pangresh J. Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. 2005. ADOPT: asynchronous distributed constraints optimizationwith quality guarantees. *Artificial Intelligence* 161:1-2 (2005), 149–180.

[24] Sofia Amador Nelke, Steven Okamoto, and Roie Zivan. 2020. Market Clearing-based Dynamic Multi-agent Task Allocation. *ACM Transactions of Intelligent Systems Technology*. 11, 1 (2020), 4:1–4:25.

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

[26] Judea Pearl. 1989. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann. I–XIX, 1–552 pages.

[27] Adrian Petcu and Boi Faltings. 2005. Approximations in Distributed Optimization. In *Proceedings of the 11th International Conference, Principles and Practice of Constraint Programming (CP)*. 802–806.

[28] Adrian Petcu and Boi Faltings. 2005. A Scalable Method for Multi-agent Constraint Optimization. In *Proceedings of the 19th International Joint Conferenceon Artificial Intelligence (IJCAI)*. 266–271.

[29] Marco Pretti. 2005. A message-passing algorithm with damping. *Journal of Statistical Mechanics: Theory and Experiment* 11 (2005), P11008.

[30] Sarvapali D. Ramchurn, Alessandro Farinelli, Kathryn S. Macarthur, and Nicholas R. Jennings. 2010. Decentralized Coordination in RoboCup Rescue. *Computer* 53, 9 (2010), 1447–1461.

[31] Alex Rogers, Alessandro Farinelli, Ruben Stranders, and Nicholas R. Jennings. 2011. Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence* 175, 2 (2011), 730–759.

[32] 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 Conferenceon Artificial Intelligence (IJCAI)*. 468–474.

[33] Pritam Som and Ananthanarayanan Chockalingam. 2010. Damped belief propagation based near-optimal equalization of severely delay-spread UWB MIMO-ISI channels. In *Proceedings of IEEE International Conference on Communications (ICC)*. 1–5.

[34] David Sontag and Tommi Jaakkola. 2009. Tree Block Coordinate Descent for MAP in Graphical Models. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 544–551.

[35] David Sontag, Talya Meltzer, Amir Globerson, Tommi Jaakkola, and Yair Weiss. 2008. Tightening LP Relaxations for MAP using Message Passing. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*. 503–510.

[36] Ruben Stranders, Alessandro Farinelli, Alex Rogers, and Nicholas R. Jennings. 2009. Decentralised coordination of continuously valued control parameters using the max-sum algorithm. In *Proceedings of the 8th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*. 601–608.

[37] Daniel Tarlow, Inmar Givoni, Richard S. Zemel, and Brendan J. Frey. 2011. Graph Cuts is a Max-Product Algorithm. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*.

[38] W. T. Luke Teacy, Alessandro Farinelli, Neil J. Grabham, Paritosh Padhy, Alex Rogers, and Nicholas R. Jennings. 2008. Max-sum decentralised coordination for sensor systems. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*. 1697–1698.

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

[40] Yair Weiss and William T. Freeman. 2001. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory* 47, 2 (2001), 736–744.

[41] Chen Yanover, Talya Meltzer, and Yair Weiss. 2006. Linear Programming Relaxations and Belief Propagation - An Empirical Study. *Journal of Machine Learning Research* 7 (2006), 1887–1907.

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

[43] William Yeoh, Ariel Felner, and Sven Koenig. 2010. BnB-ADOPT: An Asynchronous Branch-and-Bound DCOP Algorithm. *Journal of Artificial Intelligence Research (JAIR)* 38 (2010), 85–133.

[44] Zhepeng Yu, Ziyu Chen, Jingyuan He, and Yancheng Deng. 2017. A Partial Decision Scheme for Local Search Algorithms for Distributed Constraint Optimization Problems. In *Proceedings of the 16th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*. 187–194.

[45] Roie Zivan. 2008. Anytime Local Search for Distributed Constraint Optimization. In *Proceedings of the 23rd International Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*. Chicago, IL, USA, 393–398.

[46] 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.

[47] 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. *Autonomous Agents and Multi-Agent Systems* 31, 5 (2017), 1165–1207.