Max-sum Revisited: The Real Power of Damping

Liel Cohen(✉) and Roie Zivan

Industrial Engineering and Management department, Ben Gurion University, Beer-Sheva, Israel
{lielc,zivanr}@bgu.ac.il

Abstract. Max-sum is a version of Belief propagation, used for solving DCOPs. On tree-structured problems, Max-sum converges to the optimal solution in linear time. Unfortunately when the constraint graph representing the problem includes multiple cycles (as in many standard DCOP benchmarks), Max-sum does not converge and explores low quality solutions. Recent attempts to address this limitation proposed versions of Max-sum that guarantee convergence, by changing the constraint graph structure. Damping is a method that is often used for increasing the chances that Belief propagation will converge, however, it was not mentioned in studies that proposed Max-sum for solving DCOPs.

In this paper we investigate the effect of damping on Max-sum. We prove that, while it slows down the propagation of information among agents, on tree-structured graphs, Max-sum with damping is guaranteed to converge to the optimal solution in weakly polynomial time. Our empirical results demonstrate a drastic improvement in the performance of Max-sum, when using damping. However, in contrast to the common assumption, that it performs best when converging, we demonstrate that non converging versions perform efficient exploration, and produce high quality results, when implemented within an anytime framework. On most benchmarks, the best results were achieved using a high damping factor (A preliminary version of this paper was accepted as a two page extended abstract to a coming up conference.)

1 Introduction

Distributed Constraint Optimization Problem (DCOP) is a general model for distributed problem solving that has a wide range of applications in multi-agent systems. Complete algorithms for solving DCOPs [7,11] are guaranteed to find the optimal solution, but because DCOPs are NP-hard, solving optimally requires exponential time in the worst case. Thus, there is growing interest in incomplete algorithms, which may find suboptimal solutions but run quickly enough to be applied to large problems [20,21].

Whether complete or incomplete, DCOP algorithms generally follow one of two broad approaches: distributed search [7,20] or inference [2,11]. Max-sum [2] is an incomplete inference algorithm that has drawn considerable attention in recent years, including being proposed for multi-agent applications such as

© Springer International Publishing AG 2017
G. Sukthankar and J. A. Rodriguez-Aguilar (Eds.): AAMAS 2017 Visionary Papers, LNAI 10643, pp. 111–124, 2017.
https://doi.org/10.1007/978-3-319-71679-4_8

sensor systems [16]. Max-sum is actually a version of the well known Belief propagation algorithm [19], used for solving DCOPs.

Belief propagation in general (and Max-sum specifically) is known to converge to the optimal solution for problems whose constraint graph is acyclic. Unfortunately, there is no such guarantee for problems with cycles [19]. Furthermore, when the agents' beliefs fail to converge, the resulting assignments may be of low quality. This occurs because cyclic information propagation leads to inaccurate and inconsistent information being computed by the agents. Unfortunately, many DCOPs that were investigated in previous studies are dense and indeed include multiple cycles (e.g., [7]). Our experimental study revealed that on various standard benchmark problem classes (uniform and structured), Max-sum does not converge and explores low-quality solutions.

Damping is a method that was combined with Belief propagation in order to decrease the effect of cyclic information propagation. By balancing the weight of the new calculation performed in each iteration and the weight of calculations performed in previous iterations, researchers have reported success in increasing the chances for convergence of Belief propagation when applied in different scenarios [5,12,17]. Nevertheless, Damping was not mentioned in the papers that adopted Max-sum for solving DCOPs and proposed extended versions of the algorithm [2,13,22]. To the best of our knowledge there are no published indications of the effect of damping on Max-sum, when solving DCOPs.

In this paper we contribute to the development of incomplete inference algorithms for solving DCOPs by investigating the effect of using damping within Max-sum. It is important to emphasize that the contribution and novelty of this work is not in proposing the use of damping, which is a well known method that has been studied by researchers in the graphical models community (see details in Sect. 2), but rather to investigate the unique properties of this method, when applied to Max-sum in order to improve its performance when used for solving DCOPs. More specifically we make the following contributions:

1. We prove that on tree-structured graphs, Damped Max-sum converges in weakly polynomial time. This result applies to a graph with a single cycle as well (under the restrictions specified in [18]). On a directed acyclic graph structure (as used in Max-sum_ADVP [22]) the convergence is also guaranteed in weakly polynomial time, but not necessarily to the optimal solution. This result is extremely significant in distributed scenarios where agents are not aware of the global topology, only of their own neighborhood, thus, they cannot avoid the use of damping when the graph has a structure that guarantees convergence.
2. We investigate the relation between the damping factor used and the success of the damping method in improving the solutions produced by Max-sum when solving DCOPs. On most standard DCOP benchmarks, the best results were achieved for high damping factor values. However, on graph coloring problems and other problems with similar constraint structure, a high damping factor resulted in a higher convergence rate, but also in lower quality solutions.

3. We demonstrate that, in contrast to the common assumption, the best performance is achieved when Max-sum with damping does not converge, but rather performs efficient exploration that can be captured when used within an anytime framework [21]. The combination of Damped Max-sum using a high damping factor and the anytime mechanism, outperforms all other versions of Max-sum, as well as local search DCOP algorithms, on various benchmarks.

2 Related Work

The graphical models literature includes many indications for the use of damping within Belief propagation (BP). We specify a number of studies that have some resemblance to our work and from which one can learn the common assumptions regarding the effect of damping on BP.

An attempt to apply damping to BP when solving both synthetic and realistic problems, represented by Bayesian networks, was presented in [8]. The results (with a rather small damping factor, 0.1) indicated that damping reduced oscillations and increased the chances of convergence. However, in many cases the algorithm converged to inaccurate solutions (i.e., did not approximate the optimal solution well). An investigation of the relation between the damping level and the convergence rate of BP when solving K-SAT problems, was presented in [12]. Results indicated that fastest convergence is achieved for damping factor of approximately 0.5, while larger damping factors (0.9) are better for reducing oscillations.

Lazic et al. report that damping increases the chances for convergence on maximum-a-posteriori (MAP) inference problems [5]. They find that a damping factor of 0.8 is enough to achieve convergence in most cases, although their results indicate that a high damping factor may increase the number of iterations required for convergence.

The most similar study to our own seems to be [10]. For bit error problems in communication channels the effect of damping on both the convergence and the quality of the result of BP was investigated. In contrast to the results we present, they report that the method is successful in producing high quality solutions for damping factors between 0.3 and 0.7 and that the best solutions were found when using a damping factor of 0.45.

An investigation of the effect of damping on convergence of BP solving clustering data problems was presented in [1]. The results indicate that when converging, the algorithm produces similar high quality results (regardless of the damping factor) and that only for very small damping factors the algorithm does not converge (up to 0.3). Again, they report that high damping factors slow convergence.

The conclusion from this short survey is that the effect of damping on BP is highly dependent on the problem being solved. Thus, there is merit in investigating the effect of damping when solving DCOP benchmarks. Furthermore, none of the papers mentioned (and any other we know of) reports the theoretical bounds we prove or suggests the possibility that damping can be used to balance exploration and exploitation of Max-sum, as we report in this paper.

Very few studies that use Max-sum for solving DCOP report the use of damping. One such paper used a damping factor of 0.5 and found the algorithm to be inferior to standard local search algorithms [9]. There was also an attempt to use damping for local search algorithms, which is obviously less relevant to our work [6].

3 Background

3.1 Distributed Constraint Optimization

Without loss of generality, in the rest of this paper we will assume that all problems are minimization problems. Our description of a DCOP is consistent with the definitions in many DCOP studies, e.g., [7,11].

A *DCOP* is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$. $\mathcal{A}$ is a finite set of agents $\{A_1, A_2, ..., A_n\}$. $\mathcal{X}$ is a finite set of variables $\{x_1,x_2,...,x_m\}$. Each variable is held by a single agent. $\mathcal{D}$ is a set of domains $\{D_1, D_2,...,D_m\}$. Each domain D_i contains the finite set of values that can be assigned to variable x_i. An assignment of value $d \in D_i$ to x_i is denoted by an ordered pair $\langle x_i, d \rangle$. $\mathcal{R}$ is a set of relations (constraints). Each constraint $C \in \mathcal{R}$ defines a non-negative *cost* for every possible value combination of a set of variables, and is of the form $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary constraint* refers to exactly two variables and is of the form $C_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{0\}$.[1] A *binary DCOP* is a DCOP in which all constraints are binary. A *partial assignment* (PA) is a set of value assignments to variables, in which each variable appears at most once. *vars(PA)* is the set of all variables that appear in PA. A constraint $C \in \mathcal{R}$ of the form $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$ is *applicable* to PA if $x_{i_1}, x_{i_2}, \ldots, x_{i_k} \in vars(PA)$. The *cost of a PA* is the sum of all applicable constraints to PA over the assignments in PA. A *complete assignment* (or a *solution*) is a partial assignment that includes all the DCOP's variables ($vars(PA) = \mathcal{X}$). An *optimal solution* is a complete assignment with minimal cost.

For simplicity we make the standard assumptions that all DCOPs are binary DCOPs in which each agent holds exactly one variable. These assumptions are commonly made in DCOP studies, e.g., [7].

3.2 Max-Sum

[2]Max-sum operates on a *factor-graph*, which is a bipartite graph in which the nodes represent variables and constraints [4]. Each variable-node representing a variable of the original DCOP is connected to all function-nodes that represent

[1] We say that a variable is *involved* in a constraint if it is one of the variables the constraint refers to.

[2] For lack of space we describe the algorithm and its extensions briefly and refer the reader to more detailed descriptions in [2,13,22].

constraints, which it is involved in. Similarly, a function-node is connected to all variable-nodes that represent variables in the original DCOP that are involved in the constraint it represents. Variable-nodes and function-nodes are considered "agents" in Max-sum, i.e., they can send and receive messages, and perform computation.

A message sent to or from variable-node x (for simplicity, we use the same notation for a variable and the variable-node representing it) is a vector of size $|D_x|$ including a cost for each value in D_x. In the first iteration all messages include vectors of zeros. A message sent from a variable-node x to a function-node f is formalized as follows: $Q^i_{x\to f} = \sum_{f'\in F_x, f'\neq f} R^{i-1}_{f'\to x} - \alpha$, where $Q^i_{x\to f}$ is the message variable-node x intends to send to function-node f in iteration i, F_x is the set of function-node neighbors of variable-node x and $R^{i-1}_{f'\to x}$ is the message sent to variable-node x by function-node f' in iteration $i-1$. α is a constant that is reduced from all costs included in the message (i.e., for each $d \in D_x$) in order to prevent the costs carried by messages throughout the algorithm run from growing arbitrarily.

A message sent from a function-node f to a variable-node x in iteration i includes for each value $d \in D_x$: $min_{PA_{-x}} cost(\langle x, d\rangle, PA_{-x})$, where PA_{-x} is a possible combination of value assignments to variables involved in f not including x. The term $cost(\langle x, d\rangle, PA_{-x})$ represents the cost of a partial assignment $a = \{\langle x, d\rangle, PA_{-x}\}$, which is: $f(a) + \sum_{x'\in X_f, x'\neq x, \langle x', d'\rangle \in a} Q^{i-1}_{x'\to f}.d'$, where $f(a)$ is the original cost in the constraint represented by f for the partial assignment a, X_f is the set of variable-node neighbors of f, and $Q^{i-1}_{x'\to f}.d'$ is the cost that was received in the message sent from variable-node x' in iteration $i-1$, for the value d' that is assigned to x' in a. x selects its value assignment $\hat{d} \in D_x$ following iteration k as follows: $\hat{d} = \mathrm{argmin}_{\mathrm{d}\in \mathrm{D_x}} \sum_{\mathrm{f}\in \mathrm{F_x}} \mathrm{R}^{\mathrm{k}}_{\mathrm{f}\to \mathrm{x}}.\mathrm{d}$.

4 Introducing Damping into Max-Sum

A common assumption regarding Belief propagation was that it is successful when it converges, and that its main drawback is that it fails to converge on problems in which the graph used for representing them includes multiple cycles. Thus, different methods were proposed in order to guarantee the convergence of Belief propagation, e.g., by revising the optimization function, or by changing the graph structure [13,15,22].

Damping is a less radical method that was proposed for increasing the chances that Belief propagation will converge [12,14,17]. However, in contrast to the methods mentioned above, introducing damping into Belief propagation is empirically found to increase the probability of convergence, but, to best of our knowledge, there is no theoretical guarantee or even an estimation or prediction method that can identify when Belief propagation with damping will converge.

In order to add damping to Max-sum we introduce a parameter $\lambda \in (0, 1]$. Before sending a message in iteration k an agent performs calculations as in standard Max-sum. Denote by $\widehat{m^k_{i\to j}}$ the result of the calculation made by agent A_i of the content of a message intended to be sent from A_i to agent A_j in iteration k. Denote by $m^{k-1}_{i\to j}$ the message sent by A_i to A_j at iteration $k-1$. The message sent from A_i to A_j in iteration k is calculated as follows:

$$m^k_{i\to j} = \lambda m^{k-1}_{i\to j} + (1-\lambda)\widehat{m^k_{i\to j}} \tag{1}$$

Thus, λ expresses the weight given to previously performed calculations with respect to the most recent calculation performed. Moreover, when $\lambda = 0$ the resulting algorithm is standard Max-sum. We demonstrate further in this paper that a selection of a high value for λ (close to 1) increases the chances of the algorithm to converge.

In all our implementations damping was performed only by variable-nodes. This allowed us to analyze the level of damping with respect to n (the number of variables/agents).

5 Convergence Runtime Bounds

Standard Max-sum guarantees convergence in linear time to the optimal solution, when the constraint graph (and the corresponding factor-graph) is tree-structured. We first establish a weakly polynomial lower bound for this guarantee, i.e., that there exists a problem on which damping slows down the convergence by a factor of $log_{1/\lambda}(C)$, where C is the cost of the optimal solution plus ϵ and ϵ is the smallest difference between constraint costs (thus, C is the smallest possible cost for a solution, which is larger than the cost of the optimal solution). Next, we prove a (loose) upper bound on the time for convergence, which is also weakly-polynomial.

Let n be the number of variables in a problem and C as defined above.

Lemma 1. *There exists a scenario in which Max-sum with damping will converge on a tree-structured graph in no less than* $2(n-2) + log_{1/\lambda}(C)$.

Proof: Consider a factor-graph with four variable-nodes, X_1, X_2, X_3 and X_4 and three function-nodes F_{12}, F_{23} and F_{34}, as depicted in Fig. 1. Each variable has two values in its domain, a and b. All functions include infinite costs for any non equal combination of value assignments. Function F_{23} includes for both equal combinations of both variables the cost 0. Function F_{12} includes a cost of $C > \epsilon$ if both agents assign a and zero cost if they both assign b. Function F_{34} includes a cost of $C - \epsilon$ if both agents assign b and zero cost if they both assign a. Obviously the optimal solution is when all variables assign b. However, in order for variable X_4 to realize that it should assign b, X_2, which receives cost C for its value a from F_{12} in every iteration, must perform $log_{1/\lambda}(C)$ iterations before it sends a message to F_{23} with a cost larger than $C - \epsilon$ for the assignment of a. This information must path to X_4 before it can learn that it is better to assign b than a, which requires 4 sequential messages. Obviously, if we add more variables to the chain (each with two values a and b), such that the two functions adjacent to the first and last variable-nodes in the chain are identical in their costs to F_{12} and F_{34}, and all other functions are identical in costs to F_{23}, the

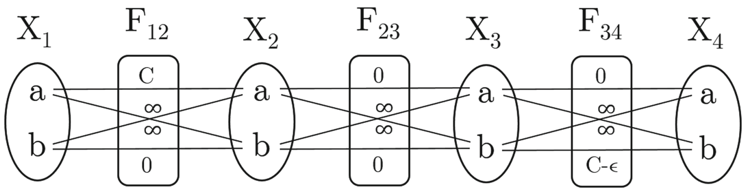

Fig. 1. Lower bound example

number of sequential messages will increase by 2 for each additional variable. Thus, a lower bound to this problem is $log_{1/\lambda}(C) + 2(n-2)$. □

Proposition 1. *The guaranteed runtime for convergence for Max-sum with damping is at least, weakly polynomial.*

Proof: An immediate corollary from Lemma 1 □

In order to produce an upper bound on the number of steps that Damped Max-sum will perform before converging to the optimal solution, we note that the convergence of standard Max-sum on tree-structured graphs is achieved in linear time because each variable-node in the factor-graph can be considered as a root of a tree to which all other agents accumulate costs (similar to a DPOP running on a pseudo-tree with no back-edges, cf. [11]). Thus, each agent, after at most a linear number of steps, knows for each of the values in its variable domain, the costs of the best solution it is involved in (for simplicity and without loss of generality, we will assume no ties).

Let n be the number of variables in a problem represented by a tree-structured factor-graph G' and $\widehat{C}$ the maximal cost sent by an agent in standard Max-sum, solving the same problem. We use η to represent the largest *ignorable* difference for a problem, i.e., the largest number such that for each cost c sent in standard Max-sum by some agent when solving the same problem, if the agent would send cost $c' = c - \eta$, the receiving agent would perform exactly the same actions as when receiving c in standard Max-sum, i.e., select the same value assignments to calculate function costs, if the receiver is a function-node, or make the same selection of value assignment, if the receiver is a variable-node.

Lemma 2. *After at most* $2(n-2)\cdot log_{1/\lambda}(\widehat{C}/\eta)$ *steps of the algorithm, a variable-node* X_i *in* G' *can select its value assignment in the optimal solution.*

Proof: Allow each of the variable-nodes, from the farthest from X_i in G' to the closest, to perform $log_{1/\lambda}(\widehat{C}/\eta)$ steps, taking into consideration only the last message received from their neighbors, and allow each function-node receiving a message to perform a single step immediately. Obviously, after these steps are completed, X_i receives costs that allow it to select its assignment in the optimal solution. □

Proposition 2. *Max-sum with Damping is guaranteed to converge to the optimal solution, on tree-structured graphs in weakly polynomial time.*

Proof: Immediate from Lemma 2. After $2(n-2) \cdot log_{1/\lambda}(\widehat{C}/\eta)$ steps, all variable-nodes can select their assignment in the optimal solution, thus, the convergence rate is polynomial in n, and $\widehat{C}/\eta$, i.e., weakly polynomial. □

We note that similar proofs can establish that Max-sum with damping produces the optimal solution on graphs with a single cycle in weakly polynomial time (subject to some restrictions [18]) and that using damping in Max-sum_AD and Max-sum_ADVP [22] slows down the convergence in each phase to, at most, weakly polynomial time.

6 Experimental Evaluation

In order to investigate the advantages of the use of damping in Max-sum, we present a set of experiments comparing different versions of the algorithm, using different λ values with standard Max-sum and two versions that guarantee convergence: Bounded_Max-sum [13] and Max-sum_ADVP [22]. We also include in our experiments the results of the well known DSA algorithm (we use type C with $p = 0.7$ [20]), in order to give an insight on the quality of the results, in comparison with local search DCOP algorithms.

We evaluated the algorithms on random uniform DCOPs and on structured and realistic problems, i.e., graph coloring, meeting scheduling and scale-free. At each experiment we randomly generated 50 different problem instances and ran the algorithms for 5,000 iterations on each of them. The results presented are an average of those 50 runs. For each iteration we present the cost of the assignment that would have been selected by each algorithm at that iteration. All algorithms were implemented within the anytime framework proposed in [21], which allowed us to report for each of them the best result it traverses within $5,000$ iterations. Also, in all versions of Max-Sum, we used value preferences selected randomly for the purpose of tie breaking, as was suggested in [2].

As mentioned above, the experiments were performed on four types of DCOPs, commonly used for evaluating DCOP algorithms, all formulated as minimization problems. Uniform random problems were generated by adding a constraint for each pair of agents/variables with probability p_1 and for each constrained pair, a cost for each combination of value assignments, selected uniformly between 1 and 10. Each problem included 100 variables with 10 values in each domain. Graph coloring problems included 50 agents and all constraints $R_{ij} \in \mathcal{R}$ were "not-equal" cost functions where an equal assignment of neighbors in the graph incurs a cost of 1 and non equal value assignments incur 0 cost. Following the literature, we used $p_1 = 0.05$ and three values (i.e., colors) in each domain [2,20,21]. Scale-free network problems included 50 agents, each holding a variable with 10 values in each domain, and were generated using the Barabási–Albert (BA) model. An initial set of 7 agents was randomly selected and connected. Additional agents were added sequentially and connected to 3 other agents with a probability proportional to the number of links per agent. Costs were independently drawn between 0 to 99. Similar problems were previously used to evaluate DCOP algorithms in [3]. Meeting scheduling problems included 90 agents, which scheduled 20 meetings into 20 time slots. When the time slots of two meetings do not allow participation in both, a cost equal to the number of agents assigned to both meetings was incurred. These realistic problems are identical to those used in [21].

Our experiments included various λ values, however, in order to avoid redundancy, we only present results with $\lambda \in 0.5, 0.7, 0.9$. This selection allows us to avoid graph density while presenting the trend of improvement of the algorithm when λ is closer to one.

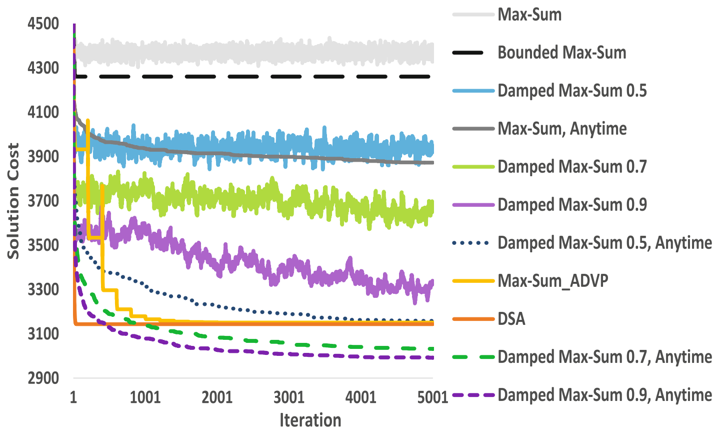

Fig. 2. Solution cost for random uniform problems with relatively low density ($p_1 = 0.1$).

Figures 2 and 3 present the solution costs found by all algorithms when solving uniform random problems containing 100 agents with a relatively low density ($p_1 = 0.1$) and with higher density of $p_1 = 0.7$ respectively. The results per iteration show that Damped Max-sum is inferior to DSA and the guaranteed convergence version Max-sum_ADVP. That been said, the anytime results of Damped Max-sum using high λ values (0.7 and 0.9) significantly outperform DSA and Max-sum_ADVP. This suggests that damping triggers efficient exploration by Max-sum, i.e., that in contrast to the assumptions made in the Belief propagation literature, the best results of Max-sum are not achieved when it converges but rather (like in the case of local search) when there is a balance between exploration and exploitation.

Figures 4, 5 and 6 present results on scale free nets, meeting scheduling and graph coloring problems, respectively. On scale free nets the trends are similar to the results obtained for uniform random problems. Damped Max-sum improves as more iterations are performed and explores solutions of higher quality. Towards the end of the run, the results per iteration of the version with $\lambda = 0.9$ produces in some iterations better solutions than DSA and similar to

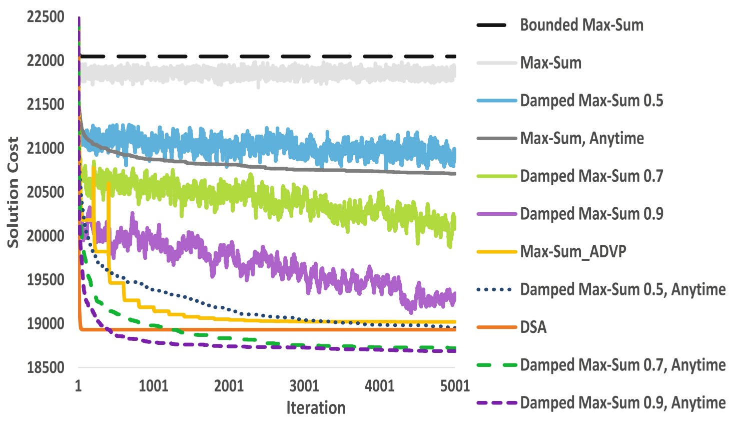

Fig. 3. Solution cost for random uniform problems with relatively high density ($p_1 = 0.7$).

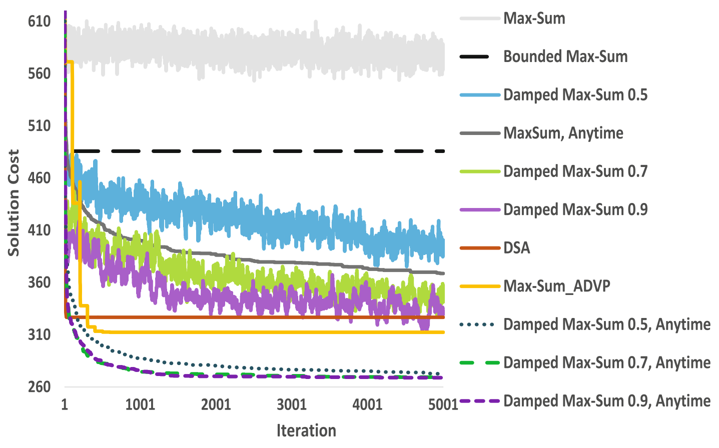

Fig. 4. Solution cost for scale free net problems.

Max-sum_ADVP. The anytime results outperform the converging algorithm significantly. On meeting scheduling and graph coloring problems, the results of the Damped Max-sum versions do not exhibit such an improvement, and seem to explore solutions of similar quality throughout the run. Interestingly, the $\lambda = 0.9$ version on graph coloring seems to perform limited exploration and traverse solutions with similar quality, while the 0.5 and 0.7 versions perform a higher level of exploration.[3]

[3] t-tests established that the Damped Max-Sum anytime solutions of all values of the parameter λ were better on average than the anytime solutions reported for standard Max-Sum, with statistical significance of $p = 0.01$, and better on average than DSA's solutions for λ values of 0.7 and 0.9 (except for the 0.9 version on graph coloring problems), with the same significance level.

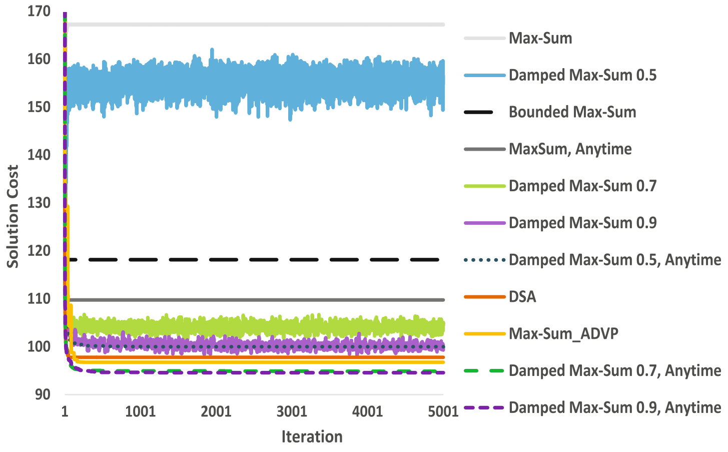

Fig. 5. Solution cost for meeting scheduling problems.

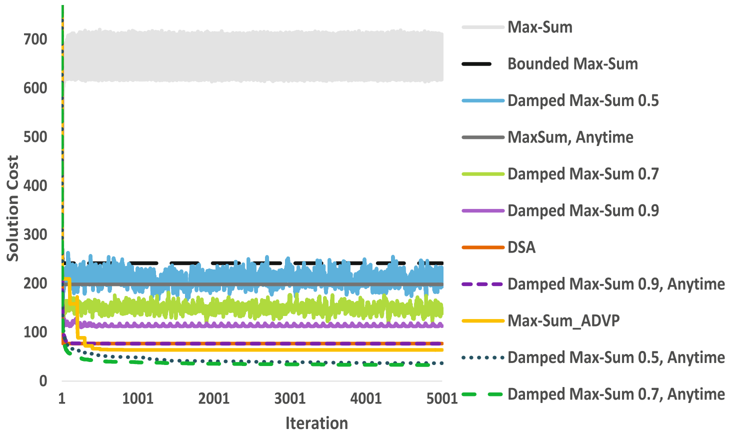

Fig. 6. Solution cost for graph coloring problems.

A closer look at Fig. 6 reveals that the $\lambda = 0.9$ version, after a small number of iterations, starts to perform limited oscillations that follow a strict pattern repeatedly. In contrast, the $\lambda = 0.5$ and $\lambda = 0.7$ versions perform rapid oscillations, which do not follow a specific pattern. Throughout the run, the average results per iteration of the $\lambda = 0.9$ version outperforms both the $\lambda = 0.5$ and $\lambda = 0.7$ versions. On the other hand, the corresponding anytime results of the $\lambda = 0.9$ version converge fast to a higher cost than the costs of the anytime solutions reported for the $\lambda = 0.5$ and $\lambda = 0.7$ versions. This is another indication of the relation between the level of exploration performed by Damped Max-sum and the quality of its anytime results.

Table 1. Convergence and anytime performance, for random uniform problems, $p_1 = 0.1$.

Problem count	Standard	0.5	0.7	0.9
Converged	0	2	7	20
% out of all 50	0%	4%	14%	40%
Anytime was better	50	48	44	33
% out of all 50	100%	96%	88%	66%

Table 2. Convergence and anytime performance, for graph coloring problems.

Problem count	Standard	0.5	0.7	0.9
Converged	1	6	8	49
% out of all 50	2%	12%	16%	98%
Anytime was better	49	44	43	37
% out of all 50	98%	88%	86%	74%

The results of our experiments indicate that, in contrast to the common assumption regarding the role of damping in improving Belief propagation, by increasing its convergence rate, the success of damping is in generating useful exploration of high quality solutions that can be captured by an anytime framework and outperform versions of Max-sum that guarantee convergence, as Max-sum_ADVP. In order to straighten this statement we present the convergence rate and anytime performance of the Max-sum versions for the uniform random problem settings and for the graph coloring problems (the convergence results of the meeting scheduling problems and the scale free nets showed similar trends to the uniform settings and were omitted for lack of space). Tables 1 and 2 present for standard Max-Sum and the Damped Max-Sum, the number of problems out of the 50 problems solved, on which each of the versions of the algorithm converged. In addition, the tables present the number of problems in which the anytime solution was better than the solution produced in the final iteration of the algorithm's run.

For the random problems (Table 1) the results indicate that the closer λ is to one, the higher are the chances of convergence of the Damped Max-Sum algorithm. The results for problem with higher density preserved the same trend and were omitted for lack of space. As for the anytime solutions reported, in problems for which the algorithm converged, it did not always converge to the best solution visited during the algorithm's run. The number of problems on which the algorithm converged to the best solution reached during the algorithm's run, increases when a higher value of λ is selected. Nevertheless, for all versions, the anytime results were better than the results in the last iteration of the algorithm on a significant portion of the problem instances.

The results in Table 2 strengthen our analysis of Fig. 6. On graph coloring problems, the $\lambda = 0.9$ has a much higher convergence rate than the $\lambda = 0.5$ and $\lambda = 0.7$ versions. However, its anytime results are better than the results in the last iteration in a fewer number of runs of the algorithm. Thus, on these problems a lower damping factor resulted in more effective exploration. In order to check whether this phenomenon was unique for graph coloring problems, we ran experiments in which we changed the constraint structure of all other benchmarks (random uniform, scale free and meeting scheduling) such that it was similar to the constraint structure in graph coloring, i.e., where for every pair of constrained variables, for each value in each domain there was a single value in the domain of the other variable with whom it was constrained. The results across all benchmarks were that the version with $\lambda = 0.9$ had higher convergence rate and produced results with higher costs than the version with $\lambda = 0.7$, as in graph coloring.

7 Conclusion

We investigated the effect of using damping within the Max-sum algorithm, the distributed version of Belief propagation, which was adopted for solving DCOPs.

In terms of computational bounds for convergence, we proved that on acyclic problems, where Max-sum is guaranteed to converge to the optimal solution, in the worst case damping slows the convergence to weakly polynomial time. Similar proofs can be applied to other structures on which Max-sum is guaranteed to converge, e.g., graphs with a single cycle and directed acyclic graphs (on which it converges, but not necessary to the optimal solution).

Our empirical study revealed that while damping improved the results of the algorithm drastically, in most cases it did not converge within 5000 iterations. However, when combined with an anytime framework, Damped Max-sum significantly outperforms the best versions of Max-sum, and a standard local search algorithm as well.

In future work we intend to deepen the investigation on the best selection of the parameter λ in Damped Max-sum, with respect to the problem structure.

References

1. Dueck, D.: Affinity propagation: clustering data by passing messages. Ph.D. thesis, University of Toronto (2009)
2. Farinelli, A., Rogers, A., Petcu, A., Jennings, N.R.: Decentralized coordination of low-power embedded devices using the max-sum algorithm. In: AAMAS, pp. 639–646 (2008)
3. Kiekintveld, C., Yin, Z., Kumar, A., Tambe, M.: Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In: AAMAS, pp. 133–140 (2010)
4. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Trans. Inf. Theor. **47**(2), 181–208 (2001)

5. Lazic, N., Frey, B., Aarabi, P.: Solving the uncapacitated facility location problem using message passing algorithms. In: International Conference on Artificial Intelligence and Statistics, pp. 429–436 (2010)
6. Verman, A.B.M., Stutz, P.: Solving distributed constraint optimization problems using ranks. In: AAAI Workshop Statistical Relational Artificial Intelligence (2014)
7. Modi, P.J., Shen, W., Tambe, M., Yokoo, M.: Adopt: asynchronous distributed constraints optimizationwith quality guarantees. Artif. Intell. **161**(1–2), 149–180 (2005)
8. Murphy, K.P., Weiss, Y., Jordan, M.I.: Loopy belief propagation for approximate inference: an empirical study. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, UAI 1999, Stockholm, Sweden, 30 July–1 August 1999, pp. 467–475 (1999)
9. Okamoto, S., Zivan, R., Nahon, A.: Distributed breakout: beyond satisfaction. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, pp. 447–453 (2016)
10. Som, A.C.P.: Damped belief propagation based near-optimal equalization of severely delay-spread UWB MIMO-ISI channels. In: 2010 IEEE International Conference on Communications (ICC), pp. 1–5 (2010)
11. Petcu, A., Faltings, B.: A scalable method for multiagent constraint optimization. In: IJCAI, pp. 266–271 (2005)
12. Pretti, M.: A message-passing algorithm with damping. J. Stat. Mech. Theor. Exp. **11**, P11008 (2005)
13. Rogers, A., Farinelli, A., Stranders, R., Jennings, N.R.: Bounded approximate decentralized coordination via the max-sum algorithm. Artif. Intell. **175**(2), 730–759 (2011)
14. Som, P., Chockalingam, A.: Damped belief propagation based near-optimal equalization of severely delay-spread UWB MIMO-ISI channels. In: 2010 IEEE International Conference on Communications (ICC), pp. 1–5. IEEE (2010)
15. Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T., Weiss, Y.: Tightening LP relaxations for map using message passing. In: UAI, pp. 503–510 (2008)
16. Stranders, R., Farinelli, A., Rogers, A., Jennings, N.R.: Decentralised coordination of mobile sensors using the max-sum algorithm. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009, Pasadena, California, USA, 11–17 July 2009, pp. 299–304 (2009)
17. Tarlow, D., Givoni, I., Zemel, R., Frey, B.: Graph cuts is a max-product algorithm. In: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (2011)
18. Weiss, Y.: Correctness of local probability propagation in graphical models with loops. Neural Comput. **12**(1), 1–41 (2000)
19. Yanover, C., Meltzer, T., Weiss, Y.: Linear programming relaxations and belief propagation - an empirical study. J. Mach. Learn. Res. **7**, 1887–1907 (2006)
20. Zhang, W., Xing, Z., Wang, G., Wittenburg, L.: Distributed stochastic search and distributed breakout: properties, comparishon and applications to constraints optimization problems in sensor networks. Artif. Intell. **161**(1–2), 55–88 (2005)
21. Zivan, R., Okamoto, S., Peled, H.: Explorative anytime local search for distributed constraint optimization. Artif. Intell. **212**, 1–26 (2014)
22. Zivan, R., Peled, H.: Max/min-sum distributed constraint optimization through value propagation on an alternating DAG. In: AAMAS, pp. 265–272 (2012)