Beyond Trees:
Analysis and Convergence of Belief Propagation in Graphs with Multiple Cycles

Roie Zivan, Omer Lev and **Rotem Galiki**
Ben-Gurion University of the Negev
Beer-Sheva, Israel
{zivanr,omerlev,rosha}@bgu.ac.il

Abstract

Belief propagation, an algorithm for solving problems represented by graphical models, has long been known to converge to the optimal solution when the graph is a tree. When the graph representing the problem includes a single cycle, the algorithm either converges to the optimal solution or performs periodic oscillations. While the conditions that trigger these two behaviors have been established, the question regarding the convergence and divergence of the algorithm on graphs that include more than one cycle is still open.

Focusing on Max-sum, the version of belief propagation for solving distributed constraint optimization problems (DCOPs), we extend the theory on the behavior of belief propagation in general – and Max-sum specifically – when solving problems represented by graphs with multiple cycles. This includes: 1) Generalizing the results obtained for graphs with a single cycle to graphs with multiple cycles, by using backtrack cost trees (BCT). 2) Proving that when the algorithm is applied to adjacent symmetric cycles, the use of a large enough damping factor guarantees convergence to the optimal solution.

Introduction

The belief propagation algorithm (Pearl 1988; Yanover, Meltzer, and Weiss 2006) is an incomplete inference (GDL-based) algorithm for solving problems that can be represented by graphical models. For example, constraint optimization (Dechter 2003), which is a general model for centralized and distributed problem solving, has a wide range of applications in artificial intelligence and multi agent systems (Ramchurn et al. 2010; Farinelli, Rogers, and Jennings 2014). In belief propagation, each node in the graph acts as an agent within a distributed algorithm, i.e., in each iteration of the algorithm it receives messages from its neighboring nodes, performs computation and sends messages to its neighbors. The agents maintain and propagate their beliefs regarding the differences in the cost (or utility)[1], which they will incur for assigning each of their possible value assignments to their variable.

Max-sum is a version of belief propagation that is used for solving distributed constraint optimization problems (DCOPs). It has drawn considerable attention in recent years, including being proposed for multi-agent applications such as sensor systems (Teacy et al. 2008; Stranders et al. 2009) and task allocation for rescue teams in disaster areas (Ramchurn et al. 2010). For convenience of presentation, we will focus on this version of belief propagation. However, all results that we present here apply in general to other versions of belief propagation as well.

Belief propagation is known to converge to the optimal solution when solving problems, which are represented by an acyclic graph. On problems represented by graphs that include cycles, the beliefs may fail to converge, and the resulting assignments that are considered optimal under those beliefs may be of low quality (Yanover, Meltzer, and Weiss 2006; Farinelli et al. 2008b; Zivan et al. 2017). This occurs because cyclic information propagation leads to inaccurate and inconsistent information being computed by the agents (Pearl 1988).

To decrease the effect of cyclic information propagation in belief propagation, the method of *damping* has been suggested, which balances the weight of the new calculation performed in each iteration and the weight of calculations performed in previous iterations. As a result, it increases the probability for convergence (Lazic, Frey, and Aarabi 2010; Som and Chockalingam 2010; Tarlow et al. 2011; Pretti 2005). Recently, splitting nodes in the factor graph on which belief propagation operates has been shown to be an effective method for triggering convergence of the algorithm, when combined with damping (Ruozzi and Tatikonda 2013; Cohen and Zivan 2018).

On graphs with a single cycle, it is known that belief propagation is not guaranteed to converge. However, when it does converge, the result is the optimal solution (Weiss 2000; Forney et al. 2001). Moreover, the conditions for convergence of belief propagation on a single cycle, are known: Forney et al. (2001) show the resemblance between the operation of the algorithm on a single cyclic graph and a chain of assignments (a route), which incurs costs with respect to

[1] In the rest of the paper we will assume, without loss of generality, that the problem is a minimization problem and therefore that violating a constraint incurs a finite cost.

the constraints along the chain. Every propagated belief is a result of a sum of costs, which are the result of the selection of value assignments in this route. Given enough iterations the route converges to a periodic traversal of the lowest cost sequence of assignments. The algorithm converges to the optimal solution if and only if this periodic route is consistent, i.e., the same variable is not assigned different values. If it does include different value assignments to the same variable, the algorithm will oscillate.

On graphs with multiple cycles, the costs (beliefs) propagated by agents are no longer the sum of constraints between assignments of a single route in the graph as in the single cycle case. Rather, they are costs accumulated through a tree, which illustrates the transfer of messages among agents (Weiss and Freeman 2001), in which each layer of nodes in the tree represents the nodes of the graph and the messages received and sent in some iteration. The layer below it includes the messages sent in the previous iteration and the layer above it the messages in the following iteration. Each layer grows with every split of routes in the graph, and thus, the weight of the costs incurred early on the beliefs may grow exponentially. Weiss and Freeman (2001) specifically announce that due to the exponential growth of costs at the bottom layers of the unwrapped tree (UT), they were not able to analyze the conditions for convergence of the algorithm as done for single cycle graphs.

In this paper we extend the theory on belief propagation in general, and particularly on Max-sum, by making the following contributions:

1. We generalize the conditions for convergence of belief propagation, presented in Forney et al. (2001) for single cycle graphs, to the general case where graphs include multiple cycles. To best of our knowledge, we are the first to offer an answer to this question that was left open by Weiss and Freeman almost two decades ago (Weiss and Freeman 2001). Inspired by the unwrapped tree of Weiss and Freeman (2001), we define a backtrack cost tree (BCT), which reveals the components that were summed in order to generate the propagated beliefs. We show that in graphs with multiple cycles, this tree implies a route of value assignments to variables. We further prove that following enough iterations, the bottom layers of all the BCTs of a variable's beliefs propagated by the algorithm are identical.

2. We prove that with a large enough damping factor, on graphs with multiple cycles, the algorithm is guaranteed to converge to the minimal value assignment route (as proven for single cycle graphs without damping by Forney et al., 2001). Thus, on graphs where the minimal route is the optimal solution, damping can be used to trigger the convergence of the algorithm to the optimal solution. We further show that in a chain (or a tree) of cycles with symmetric constraints, the minimal route is always the optimal solution. Thus, there exists a damping parameter that guarantees convergence of the algorithm to the optimal solution. This explains the high quality empirical results and fast convergence shown in Cohen and Zivan (2018) for the combination of symmetric splitting and damping.

Background

In this section we present background on the graphical models to which our results apply: the distributed constraint optimization problems (DCOPs) and the DCOP version of belief propagation – the Max-sum algorithm. The algorithm we will be discussing is actually solving a min-sum problem (as in Ruozzi and Tatikonda, 2013), but we will still refer to it as "Max-sum", since this name is commonly used (Farinelli et al. 2008a; Farinelli, Rogers, and Jennings 2014; Yedidsion, Zivan, and Farinelli 2014; Zivan et al. 2017). We will also discuss the conditions for convergence on single cycle graphs, as presented in Forney et al. (2001).

Graphical Models

Graphical models such as Bayesian networks or constraint networks are a widely used representation framework for reasoning and solving optimization problems. The graph structure is used to capture dependencies between variables (Marinescu and Dechter 2009). Our work extends the theory established in Weiss (2000), which considered the most a priori Maximum a posteriori (MAP) assignment, which is solved using the Max-product version of belief propagation. The relation between MAP and constraint optimization is well established (Marinescu and Dechter 2009; Farinelli et al. 2008a), and thus, results that consider Max-product for MAP apply to Max/Min-sum for solving constraint optimization problems, as well as the other way round (Ruozzi and Tatikonda 2013). Without loss of generality, we will focus on constraint optimization, since it is more common in AI literature. Moreover, we will consider the distributed version of the problem, since it is a natural representation for message passing algorithms. Nevertheless, our results apply to any version of problem represented by a graphical model and solved by belief propagation, as do the results of Weiss (2000).

Distributed Constraint Optimization

Without loss of generality, in the rest of this paper we will assume that all problems are minimization problems (as in many DCOP papers, e.g., Modi et al., 2005). Thus, we assume that all constraints define costs and not utilities.

A $DCOP$ is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$. $\mathcal{A}$ is a finite set of agents $\{A_1, A_2, \ldots, A_n\}$. $\mathcal{X}$ is a finite set of variables $\{X_1, X_2, \ldots, X_m\}$. Each variable is held by a single agent, and an agent may hold more than one variable. $\mathcal{D}$ is a set of domains $\{D_1, D_2, \ldots, D_m\}$. Each domain D_i contains the finite set of values that can be assigned to variable X_i. We denote an assignment of value $x \in D_i$ to X_i by an ordered pair $\langle X_i, x \rangle$. $\mathcal{R}$ is a set of relations (constraints). Each constraint $R_j \in \mathcal{R}$ defines a non-negative *cost* for every possible value combination of a set of variables, and is of the form $R_j : D_{j_1} \times D_{j_2} \times \ldots \times D_{j_k} \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary constraint* refers to exactly two variables and is of the form $R_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{0\}$.[2] For each binary constraint R_{ij} there is a corresponding cost table T_{ij} with dimensions $|D_i|$ and $|D_j|$ in which the cost in every

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

entry $e_{x,y}$ is the cost incurred when X_i is assigned x and X_j is assigned y. 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, i.e., $vars(PA) = \{X_i \mid \exists x \in D_i \wedge \langle X_i, x\rangle \in PA\}$. A constraint $R_j \in \mathcal{R}$ of the form $R_j : D_{j_1} \times D_{j_2} \times \ldots \times D_{j_k} \rightarrow \mathbb{R}^+ \cup \{0\}$ is *applicable* to PA if each of the variables $X_{j_1}, X_{j_2}, \ldots, X_{j_k}$ is included in $vars(PA)$. 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* (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 common assumption that each agent holds exactly one variable, i.e., $n = m$, and we concentrate on binary DCOPs, in which all constraints are binary. These assumptions are customary in DCOP literature (e.g., Petcu and Faltings, 2005; Yeoh, Felner, and Koenig, 2010).

The Max-Sum algorithm[3]

Max-sum operates on a *factor-graph*, which is a bipartite graph in which the nodes represent variables and constraints (Kschischang, Frey, and Loeliger 2001). Each variable-node representing a variable of the original DCOP is connected to all function-nodes that represent 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 can 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. Before the first iteration, all nodes assume that all messages they previously received (in iteration 0) include vectors of zeros. A message sent from a variable-node X to a function-node F in iteration $i \geq 1$ is formalized as follows:

$$Q^i_{X\rightarrow F} = \sum_{F' \in F_X, F' \neq F} R^{i-1}_{F'\rightarrow X} - \alpha$$

where $Q^i_{X\rightarrow F}$ 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 $R^{i-1}_{F'\rightarrow X}$ 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 (i.e., for each $x \in D_X$) in order to prevent the costs carried by messages throughout the algorithm run 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 $x \in D_X$:

$$min_{PA_{-X}} cost(\langle X, x\rangle, PA_{-X})$$

where PA_{-X} is a possible combination of value assignments to variables involved in F not including X. The term $cost(\langle X, x\rangle, PA_{-X})$ represents the cost of a partial assignment $a = \{\langle X, x\rangle, PA_{-X}\}$, which is:

$$f(a) + \sum_{X' \in X_F, X' \neq X, \langle X', x'\rangle \in a} (Q^{i-1}_{X'\rightarrow F})_{x'}$$

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 $(Q^{i-1}_{X'\rightarrow F})_{x'}$ 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:

$$\hat{x} = \arg\min_{x \in D_X} \sum_{F \in F_X} (R^k_{F\rightarrow X})_x$$

In order to add damping to Max-sum a parameter $\lambda \in [0, 1)$ is used. Before sending a message in iteration k an agent performs calculations as in standard Max-sum. Denote by $\widehat{m^k_{i\rightarrow j}}$ 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 $m^{k-1}_{i\rightarrow j}$ 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:

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

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.

Split Constraint Factor Graphs

When Max-sum is applied to an asymmetric problem, the representing factor graph has each (binary) constraint represented by two function-nodes, one for each part of the constraint held by one of the involved agents. Each function-node is connected to both variable-nodes representing the variables involved in the constraint (Zivan, Parash, and Naveh 2015).

In Ruozzi and Tatikonda (2013); Cohen and Zivan (2018) such *Split Constraint Factor Graphs* (SCFGs) are used as an enhancement method for Max-sum. This is achieved by having each constraint that was represented by a single function-node in the original factor graph, represented by two function-nodes. The SCFG is equivalent to the original factor graph, if the sum of the cost tables of the two function-nodes representing each constraint in the SCFG is equal to the cost table of the single function-node representing the same constraint in the original factor graph. By tuning the similarity between the two function-nodes representing the same constraint one can determine the level of asymmetry in the SCFG (Cohen and Zivan 2018).

[3]For lack of space we describe the algorithm briefly and refer the reader to more detailed descriptions in Farinelli et al. (2008b); Rogers et al. (2011); Zivan et al. (2017).

Single Cycle Factor Graphs

For a single cycle factor graph, we know that if belief propagation converges, it is to the optimal solution (Forney et al. 2001; Weiss 2000). Moreover, when the algorithm does not converge – it periodically changes its set of assignments. In order to explain this behavior, Forney et al. (2001) show the similarity of the performance of the algorithm on a cycle to its performance on a chain, whose nodes are similar to the nodes in the cycle, but whose length is equal to the number of iterations performed by the algorithm. One can consider a sequence of messages starting at the first node of the chain and heading towards its other end. Each message carries beliefs accumulated from costs added by function-nodes. Each function-node adds a cost to each belief, which is the constraint value of a pair of value assignments to its neighboring variable-nodes. Each such sequence of cost accumulation (route) must at some point become periodic, and the minimal belief would be generated by the minimal periodic route. If this periodic route is consistent, i.e., the set of assignments implied by the costs in it contain a single value assignment for each variable, the algorithm converges; otherwise, it does not (Forney et al. 2001). Our results generalize these insights such that similar statements can be made for any structure of the factor graph.

Preliminaries

Our analysis will include insights regarding the constructions of beliefs from costs incurred by constraints. Thus, for every pair of constrained variables, X_i and X_j, for each $x \in D_i$, $x' \in D_j$, we will denote the cost incurred by the constraint for assigning x to X_i and x' to X_j as $R(X_i = x, X_j = x')$[4].

We will be discussing the messages and assignment selections over time as iterations progress, so we shall mark the state of the algorithm at time t (that is, after t iterations) as $\bar{s}^t$. This state includes the value assignments selected by all variable-nodes in the graph at time t. The value assignments are selected according to the messages sent to the variable-nodes by their function-node neighbors (as described below).

We denote by d_{X_i} the degree of each variable node (the number of variables it is constrained with). When handling a single cycle, the degree of each node is 2, and therefore there is no need to sum anything in the message sent from the variable. However, in graphs including connected cycles, as in our running example of a lemniscate (an "8", or ∞, shape, as in Figure 1), there is at least one variable for which the degree is larger (in the case of the lemniscate, there is a node with a degree of 4). This means that messages are "amplified": As can be seen in Figure 2, a message that started as A from F_{13}, becomes 3A in the penultimate stage, and continuing it, it will then become 9A, and on and on – after i such rounds it will become 3^iA.

Moreover, while in the case of a single cycle, one could draw a message sent from a variable node and see how

[4]This is a shorthand for the $e_{x,y}$ element of the T_{ij} table representing the constraint R.

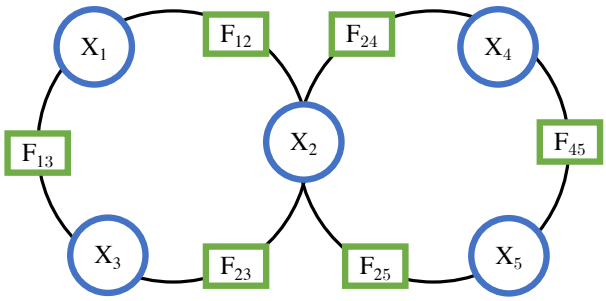


Figure 1: A lemniscate with 5 variable nodes and 6 function nodes.

it "rolls" down a path – from X_1 (through a function-node) to X_2, from there, through another function-node, to X_3, X_4, X_5 and so on, until finishing back in X_1, this is not the case with connected cycles. Here, because of the nodes with a degree over 2, the path from a variable splits and divides, resulting in a tree-like shape, with multiple routes. For example, in the lemniscate of Figure 1, a message from X_1 can pass through a function-node to X_2, from where it splits into 3 (instead of 1, in the case of a single cycle) – and from which it will continue on to the other nodes (as is detailed in Figure 2). Indeed, from each node we can draw a similar route-tree (also called an unwrapped tree (UT) by Weiss and Freeman, 2001), rooted at that node, showing how messages pass, with each level in the route-tree being one step (time wise) ahead of the level above it. Thus, at time t, we could build for any possible value assignment $x_i^t \in D_i$ to variable X_i, a route-tree that leads to this assignment. That is, a set of messages (each message is a vector, of course), whose values lead up to the choice of the possibility and availability of this particular assignment to variable X_i.

Similar to the way the route-tree is drawn from a starting point and advances with time, one can draw a mirror image of that tree. Starting from the end point – the *belief* for the cost of assigning to X_i some $x \in D_i$, as sent to a different node – the values from which that cost was calculated can be backtracked to the messages and costs added by constraints, which sum up to this belief. For example, variable X_1 got a message from constraint R_{13} (node F_{13}), which was a vector of beliefs, each of them a particular combination of a specific cost of the constraint, e.g., $R_{13}(X_1 = x, X_3 = x')$ – combined with an additional cost from the message passed to the function-node itself from X_3. That message included the costs X_3 received through F_{23}, which were themselves composed of some $R_{23}(X_2 = \bar{x}, X_3 = \tilde{x})$, along with costs received from X_2. Such backtracking can be continued until the whole expression of the value received by X_1 can be deciphered and it is known from which specific constraint cost did each component come from: every belief received by a variable can be ultimately traced to a combination of particular $R(X_i = x, X_j = x')$ costs, some of which have been multiplied and changed in the message process (see Figure 3). This backtracking creates a tree: in our example, the message passed from X_2 to F_{23} included costs X_2 received from three different nodes – F_{13}, F_{24} and F_{25}. This backtracking cost tree (BCT), that can be created for all beliefs of any variable at time t, is to be a critical element in

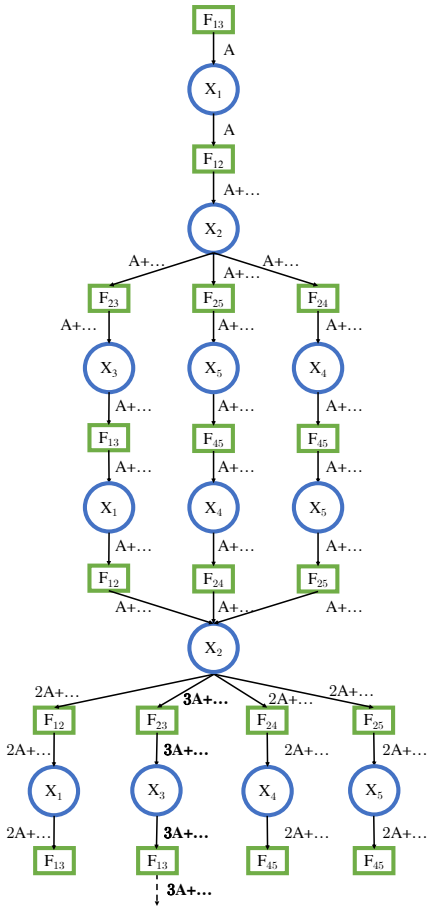


Figure 2: An example of passing messages in the lemniscate of Figure 1. Assuming a message with a value A in one of the indices was sent to X_1 from the constraint F_{13}, and that constraint values which included the A were sent in following messages (adding the A to values of those constraints), the A can be seen to grow by the time it is sent again from X_2 to F_{23}, when it is 3A+.... This is further sent to X_3 and from there to F_{13}, returning us to the beginning of the tree with a message of 3A+.... Naturally, repeating the route-tree at that point will result in the message of value 9A to be sent, and so on and on.

our analysis.

Definition 1. *A Backtracking Cost Tree (BCT) is defined for a belief that appears in a message sent from variable X_i at time t, to a function node connecting it to a variable X_j. The belief is regarding the cost of assigning some $x \in D_i$ to X_i. We shall denote it as $BCT^t_{i=x\to j}$.*

The belief, as constructed by the Max-sum algorithm, is a sum of various components, and the tree is composed from them. At the root is the belief, i.e., a cost for assigning some $x \in D_i$ to X_i, and it is connected to all nodes it received a message from at time $t-1$, with the edges containing the beliefs it was passed that ended up in the calculation of the belief it sent. Each of those nodes is connected itself to the nodes that send it messages at time $t-2$, with the edges containing the beliefs that passed to it that ended up in its message. The tree leaves are all at time 0 (see Figure 3).

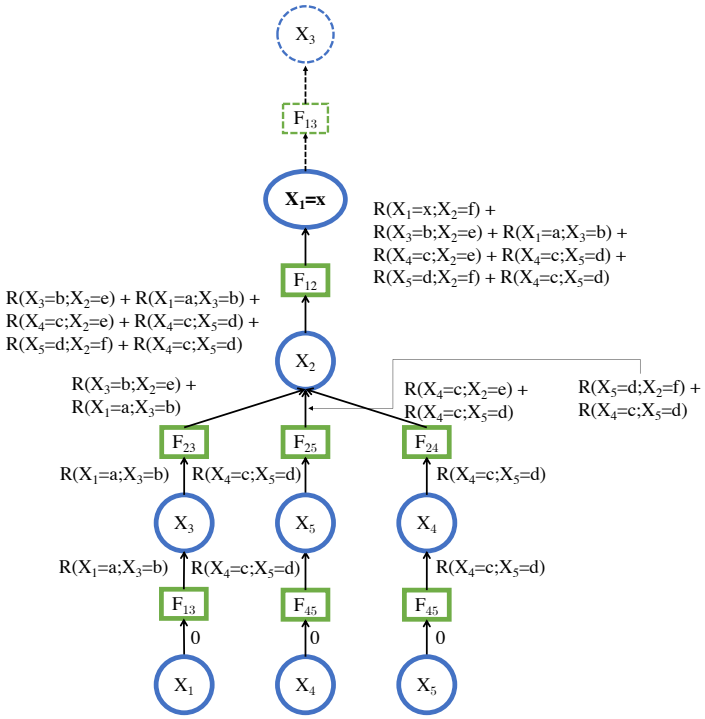


Figure 3: An example of $BCT^6_{1=x\to 3}$ in the lemniscate of Figure 1.

For each $BCT^t_{i=x\to j}$ there is an implied assignment tree, which includes the value assignments that the variables at each point of the tree would have to assign to incur the cost 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 this variable-node in iteration $t-1$. We shall mark the tree for this minimal sum of beliefs as BCT^t_i, as it doesn't depend on any specific belief which appears in a message to a different variable.

Max-sum and BCT

We begin by illuminating some of the properties of Max-sum on any graph. Some of these properties will be used in the proofs of our main results in the following section:

Observation 1. *For any factor graph G, at any time $t \in \mathbb{N}$, in any message sent, there exists a belief which is minimal.*[5]

When evaluating the cost of a sequence of states, we wish to evaluate its average cost, not its absolute value. So a sequence of 5 states which add a cost of 10 is not as costly as a sequence of 2 states which add a cost of 5. The costs are normalized by their length.

Lemma 1. *For any factor graph G, there is a time $t_0 \in \mathbb{N}$ and a positive integer k such that for any $t \geq t_0$, the state $\vec{s}^t = \vec{s}^{t+k}$. That is, the variable assignments and the costs added to the BCTs are periodic. We call this periodic*

[5]For simplicity we assume there are no ties. Ties can be avoided using methods as suggested in Farinelli et al. (2008a).

sequence, which is the endstate of the Max-sum process, the final periodical[6]. *This final periodical is one for which the assignment BCT tree for every variable X_i generated by this process – BCT_i^t – is minimal.*

Proof. The number of possible assignments is finite. Therefore a state $\vec{s}$ appears infinitely many times, and there are sequences of states between two different appearances of $\vec{s}$, which appear infinitely many times. Let us assume for contradiction there are two different such infinitely occurring sequences. One of these sequences has to be the one for which the (average) cost is not minimal, and we shall denote it as A. That means there is a different sequence (possibly of different length) – which we shall denote as B – with a smaller cost. Since the non-minimal sequence repeats endlessly, that means that in A there is at least one variable X_i for which the beliefs for a particular assignment (say, $X_i = y$) are smaller than they are for the assignment of the minimal sequence (which is $X_i = x$). Suppose the difference between these two values is c.

Since the costs are cumulative in Max-sum, this means that after enough steps (at least $|A| \cdot |B| \left\lceil \frac{c}{A-B} \right\rceil$ occurrences of the A sequence), the cost for X_i for x is now smaller than for assigning y, and now y will never be chosen in the future, contradicting A's infinite occurrence. Thus there is only one such infinitely occurring sequence, it is minimal, and the time when it starts to occur endlessly is denoted t_0. □

Lemma 2. *Let X_i be a variable located on a cycle (i.e., there is a route from X_i to itself). There is a time $t_1 \in \mathbb{N}$ ($t_1 \geq t_0$) and $q \in \mathbb{N}$, such that the BCTs of all beliefs in a message from X_i have the same q bottom layers (that is, furthest away from the root, which is the current belief).*

Proof. Denote by $BCT^t_{i=x\to j}$ the BCT for the minimal belief in the message sent by X_i at time t during the endstate periodic sequence (i.e., $t > t_0$) to X_j, and let $BCT^t_{i=y\to j}$ be the BCT for some other belief in that same message. Let us denote the route from X_i back to itself through X_j with the minimal assignment $X_i = x$ as route A, and the route through assigning y to X_i as B. Let c be the difference in cost between these two routes.

Every time $|A| \cdot |B|$ steps through all states are completed, the difference between $BCT_{i=x\to j}$ and $BCT_{i=y\to j}$ only grows, since the cost for the routes accumulate, and we know $BCT_{i=x\to j}$ is the minimal belief for X_i at time t. Let us denote this difference at time t by u. Given enough steps, there is a number $t' \in \mathbb{N}$, such that $t' \cdot c > u$. Thus, after t' steps, it would no longer be beneficial to chose the belief based on $X_i = y$, but rather the one based on $X_i = x$. Hence, for $t_1 > t_0 + t'$, the lemma holds for that part of the final periodical (a similar process can be done for any message in the periodic sequence, as it has a finite length). □

Corollary 1. *For any pair of variable-nodes X_i and X_k which reside on the same cycle in the factor graph, then there is a time $t_1 \in \mathbb{N}$ ($t_1 \geq t_0$) and $q \in \mathbb{N}$, such that the q bottom layers (that is, furthest away from the root) of $BCT^t_{i=x\to j}$ are contained in (for any $x \in D_i$) those of $BCT^t_{k=y\to l}$ (for any $y \in D_k$ and X_l which X_k is connected to through a function-node) at any time $t > t_1$.*

Proof. Since all beliefs in the message from X_i have the same q bottom layers (from Lemma 2), any $BCT^t_{k=y\to l}$, since it is building itself based on a message received from X_i, is basing its own BCT on at least one of those beliefs, and since all have the same bottom q layers, $BCT^t_{k=y\to l}$ includes them. □

Damping and Convergence

Thanks to Lemma 1, we know there is an appropriate t_0 from which the final sequence begins, and the assignments are periodical. We use this to also define, $\overline{BCT}^t_{i=x\to j}$, which is like $BCT^t_{i=x\to j}$, but instead of culminating in time 0 (that is, all leaves are at time $t = 0$), culminates at time t_0. In a similar way, $\overline{BCT}^t_i$ is the equivalent of BCT^t_i which starts at t_0. When using a damping factor λ we shall denote the BCT as λ-BCT.

Theorem 1. *When damping is used, for a large enough damping factor λ, the endstate will be determined by λ-$\overline{BCT}^t_{i=x\to j}$ (for each $x \in D_i$).*

Proof. Once choosing to use the Max-sum algorithm with damping, costs and message values change compared to the no damping case. However, for the same reasoning as in Lemma 1, the algorithm will still reach, an endstate at some time t_0 when using damping as well (note that this time does not have to be the same time a Max-sum algorithm without damping would reach the endstate, and the BCT itself is different for different λ values as well).

Let ϵ be the smallest possible difference between two constraint values possible in G, and d be the maximal degree in G. We shall denote the cost of the beliefs in $BCT^{t_0}_{i=x\to j}$, if we summed them up without damping, as c. The maximal possible value that can appear in a cost of a belief for Max-sum in $BCT^t_{i=x\to j}$ without damping is $(d-1)^{t-t_0}c$.

Please recall that λ is the damping factor, and that $(1-\lambda)$ is the multiplicative coefficient applied to the non-damped message. Let us now take $1 - \lambda < \frac{1}{2d}$. There is a $k' \in \mathbb{N}$ such that $\frac{1}{2}^{k'} c < \frac{\epsilon}{2}$. Thus, looking at λ-BCT at time $t = t_0 + k$ for $k > k'$, any influence by c is going to be $\leq (1-\lambda)^{t-t_0}(d-1)^{t-t_0}c \leq \frac{1}{2}^{t-t_0} c \leq \frac{\epsilon}{2}$.

Thus, any decision on which belief to choose will not be affected by any value that occurred in λ-$BCT^{t_0}_{i=x}$, since no value of size $\frac{\epsilon}{2}$ can have any influence on the choice of belief that nodes make. Since the endstate started at t_0, and is still ongoing in time $t_0 + k$, this means the endstate is not influenced by anything that went on in the initial t_0 stages, and only has to do with λ-$\overline{BCT}^t_{i=x\to j}$. □

Before stating the corollary of our main theorem, note again that every $BCT^t_{i=x\to j}$ (including λ-$BCT^t_{i=x\to j}$ for any $\lambda \in [0,1)$) induces an equivalent assignment tree – what

[6]Note that when Max-sum reaches convergence, $k = 1$, as its values do not change.

would be the assignment of a variable at each point given the messages it sent. These assignments may be consistent, i.e., they stay the same (the same values are assigned to the same variables at different times) and are not contradictory, and sometimes they are not.

Corollary 2. *For a large enough damping factor λ, if the assignment tree induced by λ-$\overline{BCT}_i^t$ is consistent, then Max-sum with a λ damping factor will converge to the optimal solution.*

Proof. Thanks to Theorem 1, we know we do not need to care regarding λ-BCT_i^t, but only regarding λ-$\overline{BCT}_i^t$. If that is consistent, then we know we have converged. Thanks to Lemma 1, we know that we have converged to a minimal value state (and minimal in average cost is minimal in absolute cost in this case, since the sequence length is 1). □

It is important to notice that Corollary 2 does not imply that whenever Max-sum converges it is to the optimal solution. There is plenty of empirical evidence that when Max-sum is combined with damping it converges to sub-optimal solutions. That is because the damping factor may not be large enough, and hence it does not eliminate the effect of the costs accumulated before t_0, but only reduces their influence on BCT_i^t. And thus, a periodical may include a consistent assignment, which is not the optimal.

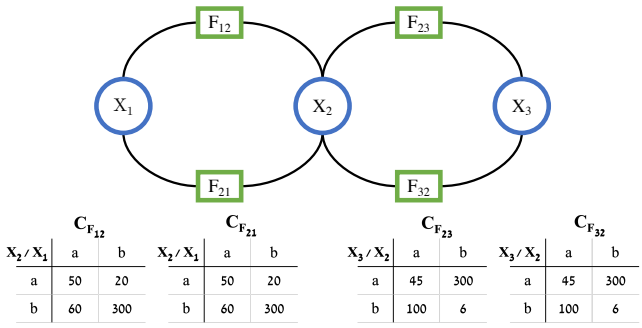
Theorem 1 and Corollary 2 can help us explain recent empirical results obtained when including both damping and splitting in Max-sum: The results presented by Cohen and Zivan (2018) indicate that damped Max-sum on symmetric SCFGs, converges very fast to high quality solutions. For specific graph structures we can even reach a more specific result:

Proposition 1. *In any SCFG with a linear division[7] of function nodes, if its original graph was a tree, then for a high enough damping factor λ, the assignment induced by λ-$\overline{BCT}_i^t$ is consistent and optimal.*

Proof. Following Corollary 2, we know that for large enough λ, it is enough to show that λ-$\overline{BCT}_i^t$ is consistent. Since in an SCFG, each constraint is represented by two function-nodes, and since the original graph was a tree, there is no other cycle in the factor-graph besides the cycles generated by split. The only possible inconsistent route must include for some pair of constrained variables X_i and X_j, two different pairs of value assignments, which one of them incurs a smaller cost in F_{ij} and the other incurs a smaller cost in F_{ji}. This contradicts the linear division of the SCFG. □

Example 1. *The example depicted in Figure 4 demonstrates the effect of damping when applying Max-sum to the type of graphs, which Proposition 1 addresses (here, the split is symmetric).*

[7] A linear division of an SCFG is one where for every function node F_{ij}, there is a constant $q \in (0, 1)$ such that for each entry $z \in F_{ij}$ the ratio between the corresponding entries in the split function-nodes is q.



$C_{F_{12}}$

X_2/X_1	a	b
a	50	20
b	60	300

$C_{F_{21}}$

X_2/X_1	a	b
a	50	20
b	60	300

$C_{F_{23}}$

X_3/X_2	a	b
a	45	300
b	100	6

$C_{F_{32}}$

X_3/X_2	a	b
a	45	300
b	100	6

Figure 4: Two adjacent constraint SCFG Example

The optimal solution is $X_1 = b$, $X_2 = a$, $X_3 = a$, with cost 130. The pair of the partial assignments $X_1 = a$, $X_2 = a$ incurs a cost of 50, and the lowest cost complete assignment that includes it is $X_1 = a$, $X_2 = a$, $X_3 = a$ with cost 190. However, in the first iteration the messages sent to X_1 includes costs of 50 for a. This cost is included in the bottom layers of the BCTs of beliefs propagated by the algorithm exponentially many times. Thus, the algorithm fails to converge. However, when using damping with $\lambda = 0.7$ the algorithm converges to the optimal solution after 23 iterations. On the other hand, if we use a smaller damping factor ($\lambda = 0.3$), the algorithm will converge to the suboptimal solution $X_1 = a$, $X_2 = b$, $X_3 = b$ with cost 132 after 19 iterations.

Conclusion

Belief propagation is a well known and widely used algorithm for solving combinatorial optimization problems that can be represented by a graphical model. Until now, the ability to analyze and characterize the convergence of the algorithm to stable states (and optimal ones), was known only for graphs which are either trees or have a single cycle. In this paper, we extend our knowledge to more graph shapes, and introduce a new analytical tool. Focusing on Max-sum, the version of belief propagation solving a distributed constraint optimization problem, we generalize the results obtained almost two decades ago for single cycle graphs to binary constraint graphs with any structure. Our analysis makes use of BCTs, a tree structured component we defined, which captures the components of the belief costs propagated by the nodes in the graph.

Our main result sheds light on how using damping in combination with the Max-sum algorithm can help trigger convergence. Moreover, our analysis illuminates recent empirical results, which used Max-sum with damping and split function-nodes, and obtained very fast convergence to high quality solutions.

We believe that using BCTs – in particular, in combination with damping – can help open the doors to further analysis of convergence and optimality in more complex graph structures. Moreover, it can trigger the design of heuristics that will exploit these properties and produce high quality solutions in short time.

Acknowledgment: This research was partially supported by the Israeli Ministry of Science and Technology.

References

Cohen, L., and Zivan, R. 2018. Balancing asymmetry in max-sum using split constraint factor graphs. In *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, 669–687.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufman.

Farinelli, A.; Rogers, A.; Petcu, A.; and Jennings, N. 2008a. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, 639–646. International Foundation for Autonomous Agents and Multiagent Systems.

Farinelli, A.; Rogers, A.; Petcu, A.; and Jennings, N. R. 2008b. Decentralized coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS*, 639–646.

Farinelli, A.; Rogers, A.; and Jennings, N. R. 2014. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Autonomous Agents and Multi-Agent Systems* 28(3):337–380.

Forney, G. D.; Kschischang, F. R.; Marcus, B.; and Tuncel, S. 2001. Iterative decoding of tail-biting trellises and connections with symbolic dynamics. In Marcus, B., and Rosenthal, J., eds., *Codes, Systems, and Graphical Models*. Springer. 239–264.

Kschischang, F. R.; Frey, B. J.; and Loeliger, H. A. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47:2:181–208.

Lazic, N.; Frey, B.; and Aarabi, P. 2010. Solving the uncapacitated facility location problem using message passing algorithms. In *International Conference on Artificial Intelligence and Statistics*, 429–436.

Marinescu, R., and Dechter, R. 2009. AND/OR branch-and-bound search for combinatorial optimization in graphical models. *Artif. Intell.* 173(16-17):1457–1491.

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

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, California: Morgan Kaufmann.

Petcu, A., and Faltings, B. 2005. A scalable method for multiagent constraint optimization. In *IJCAI*, 266–271.

Pretti, M. 2005. A message-passing algorithm with damping. *Journal of Statistical Mechanics: Theory and Experiment* 11:P11008.

Ramchurn, S. D.; Farinelli, A.; Macarthur, K. S.; and Jennings, N. R. 2010. Decentralized coordination in robocup rescue. *Computer J.* 53(9):1447–1461.

Rogers, A.; Farinelli, A.; Stranders, R.; and Jennings, N. R. 2011. Bounded approximate decentralized coordination via the max-sum algorithm. *Artificial Intelligence* 175(2):730–759.

Ruozzi, N., and Tatikonda, S. 2013. Message-passing algorithms: Reparameterizations and splittings. *IEEE Trans. Information Theory* 59(9):5860–5881.

Som, P., and Chockalingam, A. 2010. Damped belief propagation based near-optimal equalization of severely delay-spread uwb mimo-isi channels. In *Communications (ICC), 2010 IEEE International Conference on*, 1–5. IEEE.

Stranders, R.; Farinelli, A.; Rogers, A.; and Jennings, N. R. 2009. Decentralised coordination of mobile sensors using the max-sum algorithm. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 299–304.

Tarlow, D.; Givoni, I.; Zemel, R.; and Frey, B. 2011. Graph cuts is a max-product algorithm. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*.

Teacy, W. T. L.; Farinelli, A.; Grabham, N. J.; Padhy, P.; Rogers, A.; and Jennings, N. R. 2008. Max-sum decentralized coordination for sensor systems. In *AAMAS*, 1697–1698.

Weiss, Y., and Freeman, W. T. 2001. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Trans. Information Theory* 47(2):736–744.

Weiss, Y. 2000. Correctness of local probability propagation in graphical models with loops. *Neural Computation* 12(1):1–41.

Yanover, C.; Meltzer, T.; and Weiss, Y. 2006. Linear programming relaxations and belief propagation - an empirical study. *Journal of Machine Learning Research* 7:1887–1907.

Yedidsion, H.; Zivan, R.; and Farinelli, A. 2014. Explorative max-sum for teams of mobile sensing agents. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, 549–556.

Yeoh, W.; Felner, A.; and Koenig, S. 2010. Bnb-adopt: An asynchronous branch-and-bound dcop algorithm. *Artificial Intelligence Research (JAIR)* 38:85–133.

Zivan, R.; Parash, T.; Cohen, L.; Peled, H.; and Okamoto, S. 2017. Balancing exploration and exploitation in incomplete min/max-sum inference for distributed constraint optimization. *Autonomous Agents and Multi-Agent Systems* 31(5):1165–1207.

Zivan, R.; Parash, T.; and Naveh, Y. 2015. Applying max-sum to asymmetric distributed constraint optimization. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 432–439.