Message Passing Algorithms for Semiring-Based and Valued Constraint Satisfaction Problems

Hong Xu, Cheng Cheng, Sven Koenig, and **T. K. Satish Kumar**
University of Southern California, Los Angeles, California 90007, United States of America
{hongx, chen260, skoenig}@usc.edu, tkskwork@gmail.com

Abstract

Local consistency algorithms, like arc consistency (AC) algorithms, are polynomial-time algorithms that prune the search space of constraint satisfaction problems (CSPs). In this paper, we present connections between message passing algorithms and AC for semiring-based CSPs (SCSPs) and valued CSPs (VCSPs), two well-established frameworks that generalize CSPs. Message passing algorithms are well known distributed search algorithms for solving many combinatorial problems in artificial intelligence, probabilistic reasoning, and information theory. However, the relationship between message passing algorithms and SCSPs or VCSPs still remains understudied. Towards this end, we propose the best-$\odot$ message passing (BOMP) algorithm for SCSPs and VCSPs. We prove that, unlike other standard message passing algorithms which are in general not guaranteed to converge, the BOMP algorithm guarantees convergence for SCSPs and specific subclasses of VCSPs. We also theoretically study the relationship between the BOMP algorithm and AC on SCSPs, and empirically study the quality of the solutions produced by the BOMP algorithm for VCSPs.

Introduction

Crisp constraint satisfaction problems (crisp CSPs), also known as *classical CSPs*, as *hard CSPs*, or simply as CSPs, are representationally powerful and have been used to solve many real-world combinatorial problems, such as map coloring and job-shop scheduling (Bistarelli et al. 1999). Crisp CSPs are known to be NP-hard in general (Bistarelli et al. 1999). Crisp CSPs are defined by a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ is a set of variables; $\mathcal{D}$, the *domain* of the crisp CSP, is a function that maps a variable X_i to its discrete domain $\mathcal{D}(X_i)$; and $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ is a set of constraints. Each C_i consists of a subset $S(C_i)$ of $\mathcal{X}$ and a list of allowed assignments of values to these variables chosen from their domains. The task in solving the crisp CSP is to find an assignment of values to all variables in $\mathcal{X}$ such that all constraints are satisfied by the assignment, i.e., all constraints allow the assignment. This assignment is called a *solution* of this crisp CSP.

Local consistency of crisp CSPs is a class of properties over subsets of variables. A crisp CSP is said to be k-*consistent* iff, for any subset of $(k-1)$ variables, any consistent assignment of values to them (i.e., all constraints among them are satisfied) can be extended to any other variable, i.e., there exists an assignment of a value to this variable that is consistent with the $(k-1)$ variables. Local consistency has been studied for its theoretical as well as practical usefulness in solving crisp CSPs. On the practical side, enforcing local consistency prunes the search space. On the theoretical side, enforcing strong k-consistency solves a crisp CSP if k is greater than or equal to the treewidth of the crisp CSP (Freuder 1982). *Arc consistency* (AC) is k-consistency where $k = 2$. It is the most common form of k-consistency that is used to prune the search space of a crisp CSP. In addition, enforcing AC is also known to solve crisp CSPs with only max-closed constraints (Jeavons and Cooper 1995).

Despite the representational power of crisp CSPs, many real-world problems require non-crisp representations, such as uncertainty and continuous variables. To overcome these limitations, many generalizations of crisp CSPs have been developed by researchers across different fields. These include weighted CSPs (WCSPs), fuzzy CSPs, probabilistic CSPs, and lexicographic CSPs. They have been used to solve many real-world problems, such as locating motifs in RNAs in molecular biology (Zytnicki, Gaspin, and Schiex 2008), finding ground states of spin glasses in statistical physics (Mézard and Montanari 2009), energy minimization in computer vision (Kolmogorov 2005), as well as the max-a-posteriori (MAP) problem in probabilistic reasoning (Koller and Friedman 2009). To unify these generalizations of crisp CSPs, (Bistarelli et al. 1999) developed two frameworks, namely semiring-based and valued CSPs (SCSPs, VCSPs). Therefore, a study of SCSPs and VCSPs is beneficial, since algorithms developed for them can be adapted easily to various generalizations of crisp CSPs. AC and other types of local consistency in crisp CSPs have also been generalized to SCSPs and VCSPs, such as *weighted arc consistency* (WAC) (Larrosa and Schiex 2004), *full directional arc consistency* (FDAC) (Larrosa and Schiex 2003), and so on, with their significance carried over.

Message passing algorithms, a class of distributed search algorithms based on processing and passing local information, have been successfully applied to solve many combinatorial problems, such as the minimum vertex cover problem (Xu et al. 2018) and distributed combinatorial opti-

mization problems (Farinelli et al. 2008; Fioretto et al. 2018). They have also been used as theoretical tools to study fundamental combinatorial problems, such as the minimum (weighted) vertex cover problem (Weigt and Zhou 2006; Nakajima et al. 2018) and the k-satisfiability problem (Mézard and Zecchina 2002). Although a complete theoretical analysis of the convergence and correctness of message passing algorithms is elusive, they work well in practice on many important combinatorial problems.

Despite the individual significance of message passing algorithms and SCSPs/VCSPs, their relationship remains understudied. In this paper, we propose a message passing algorithm for SCSPs and VCSPs, called the *best-$\odot$ message passing* (BOMP) algorithm. We prove that, unlike other standard message passing algorithms, the BOMP algorithm always converges in polynomial time for SCSPs and specific subclasses of VCSPs. We also prove that the BOMP algorithm produces solutions to SCSPs that are always arc consistent while other standard message passing algorithms in general do not produce solutions with explicit properties. Finally, we empirically study the solutions produced by the BOMP algorithm for general VCSPs. Through this paper, we intend to bring search techniques used in the probabilistic reasoning and constraint reasoning communities closer to each other.

Notes on our contributions An algorithm similar to the BOMP algorithm for SCSPs has been studied under a different formulation (Werner 2015). However, in this paper, we prove properties with respect to AC more explicitly, rather than using marginal consistency, and therefore have more and stronger properties proven specifically for arc consistency. For example, we prove that the BOMP algorithm converges in polynomial time, while (Werner 2015) only proves that its message passing algorithm reaches a fixed point in a finite number of steps (in fact, it does not converge in polynomial time in general); in addition to what (Werner 2015) has proved, we also prove that the fixed point produced by the BOMP algorithm preserves all solutions (Theorem 3). Unlike (Kolmogorov 2006), the BOMP algorithm is a simple message passing algorithm without any additional encapsulation, and therefore our results are more general.

Background

Semiring-Based CSPs

SCSPs are semiring-based generalizations of crisp CSPs. Although they were first introduced by (Bistarelli et al. 1999), we define SCSPs and their related concepts using a simpler equivalent formalism as follows (which suffices for the purposes of this paper).

Definition 1. A *semiring* is defined as a tuple $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that

- A is a set and $\mathbf{0}, \mathbf{1} \in A$;
- $+$ (additive operator) is a closed (i.e., $\forall a, b \in A : a + b \in A$), commutative (i.e., $\forall a, b \in A : a + b = b + a$) and associative (i.e., $\forall a, b, c \in A : a + (b + c) = (a + b) + c$) operator such that $\forall a \in A : a + \mathbf{0} = \mathbf{0} + a = a$,
- $\times$ (multiplicative operator) is a closed and associative operator such that $\forall a \in A : (\mathbf{1} \times a = a \times \mathbf{1} = a) \wedge (a \times \mathbf{0} = \mathbf{0} \times a = \mathbf{0})$, and
- $\times$ distributes over $+$, i.e., $\forall a, b, c \in A : a \times (b + c) = (a \times b) + (a \times c)$.

Definition 2. A *c-semiring* is a semiring in which $+$ is idempotent (i.e., $\forall a \in A : a + a = a$), $\times$ is commutative and $\forall a \in A : a + \mathbf{1} = \mathbf{1} + a = \mathbf{1}$. The partial order $\leqslant_S$ is defined as: $a \leqslant_S b$ iff $a + b = b$. Here, a is said to be *worse* than b, and b is said to be *better* than a. $a <_S b$ iff $a \leqslant_S b$ and $a \neq b$. a_1 is called a *best* of $B = \{a_1, a_2, \ldots, a_N\}$ iff $\forall a \in B : a_1 \not<_S a$. $\mathrm{best}[B]$ is defined as $a_1 + \cdots + a_N$. ($\mathrm{best}[B]$ is therefore a best of $B \cup \{\mathrm{best}[B]\}$.)

Definition 3. An SCSP P is defined as a tuple $\langle S, \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$, where

- S is a c-semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$,
- $\mathcal{X}$ is a set of variables,
- $\mathcal{D}$ is a function that maps each variable $X_i \in \mathcal{X}$ to its finite discrete domain $\mathcal{D}(X_i)$, and
- $\mathcal{C}$ is a set of constraints. Each constraint $C \in \mathcal{C}$ is an ordered pair $\langle def, Y\rangle$, where
  - $Y = \{Y_1, \ldots, Y_{|Y|}\} \subseteq \mathcal{X}$, denoted by $S(C)$, is a subset of all variables, and
  - def, denoted by $E_C(\cdot)$, is a function $\mathcal{D}(Y_1) \times \cdots \times \mathcal{D}(Y_{|Y|}) \to A$.

Definition 4. Given an SCSP $P = \langle S, \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$, a *solution* sol is an assignment of values to all variables, i.e., a function that maps each variable $X_i \in \mathcal{X}$ to its domain $\mathcal{D}(X_i)$. A solution sol is *consistent* iff $\forall C \in \mathcal{C} : \mathbf{0} <_S E_C(sol|S(C))$, where $sol|S(C)$ is the assignment of values to variables in $S(C)$ that is consistent with sol. The total valuation (weight) of sol in P is defined as $W(sol) = \times_{C \in \mathcal{C}} E_C(sol|S(C))$. A solution sol is optimal iff for any solution sol', $W(sol) \not<_S W(sol')$ holds.

Definition 5. Given an SCSP $P = \langle S, \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$, a subset of $(k-1)$ variables $Y = \{Y_1, \ldots, Y_{k-1}\} \subseteq \mathcal{X}$ is said to be *k-consistent* with respect to a k^{th} variable Y_k iff for any assignment a of values to all variables in Y, $\times_{C_j \in \{C \in \mathcal{C} \mid S(C) \subseteq Y\}} E_{C_j}(a|S(C_j)) \leqslant_S +_{y_k \in \mathcal{D}(Y_k)} \big[\times_{C_j \in \{C \in \mathcal{C} \mid S(C) \subseteq Y \cup \{Y_k\}\}} E_{C_j}(a \cup \{Y_k = y_k\}|S(C_j))\big]$ holds. P is said to be k-consistent iff each set of $(k-1)$ variables is k-consistent with respect to any k^{th} variable. 2-consistency is also called AC, i.e., a variable X_i is arc consistent with respect to another variable X_k iff, for all $x_i \in \mathcal{D}(X_i)$, $\times_{C_j \in \{C \in \mathcal{C} \mid S(C) = \{X_i\}\}} E_{C_j}(\{X_i = x_i\}) \leqslant_S +_{x_k \in \mathcal{D}(X_k)} \big[\times_{C_j \in \{C \in \mathcal{C} \mid S(C) \subseteq \{X_i, X_k\}\}} E_{C_j}(\{X_i = x_i, X_k = x_k\}|S(C_j))\big]$ holds.

Crisp CSPs can be seen as SCSPs under many different specializations. For example, a crisp CSP can be seen as an SCSP in which (a) $A = \{\mathbf{0}, \mathbf{1}\}$, where $\mathbf{0}$ and $\mathbf{1}$ are Boolean False and True, respectively, (b) $+$ and $\times$ are the OR and AND operators, respectively, (c) in each constraint

$C = \langle def, Y \rangle$, def maps disallowed assignments of values to $\mathbf{0}$ and allowed assignments of values to $\mathbf{1}$, and (d) a solution to the crisp CSP is a consistent solution.

Valued CSPs

VCSPs, first introduced by (Bistarelli et al. 1999), are alternative generalizations of crisp CSPs. They annotate each constraint with a *valuation* (weight) to denote its impact. As before, we define VCSPs and their related concepts using a simpler equivalent formalism as follows.

Definition 6. A *valuation structure* is defined as a tuple $\langle E, \circledast, \prec, \top, \bot \rangle$, such that

- E is a set totally ordered by $\prec$ with a maximum element $\top$ and a minimum element $\bot$; its elements are called valuations (weights);
- $\circledast$ is a closed, commutative, and associative binary operator on E that satisfies
  - identity (i.e., $\forall a \in E : a \circledast \bot = a$) and
  - monotonicity (i.e., $\forall a, b, c \in E : (a \preceq b) \implies (a \circledast c) \preceq (b \circledast c)$).

a_1, denoted by $\mathrm{best}[B]$, is called the *best* of $B = \{a_1, a_2, \ldots, a_N\}$ iff $\forall a \in B : a_1 \preceq a$. In this definition, $\top$ corresponds to a completely unacceptable violation and can therefore be used to express "hard" constraints. $\bot$, on the other hand, corresponds to complete satisfaction. $\forall a \in E : a \circledast \top = \top$ always holds, since $\forall a \in E : \top = (\top \circledast \bot) \preceq (\top \circledast a)$ and $\forall a \in E : \top \circledast a \preceq \top$.

Definition 7. A VCSP P is defined as a tuple $\langle S, \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $S = \langle E, \circledast, \prec, \top, \bot \rangle$ is a valuation structure, $\mathcal{X}$ is a set of variables, $\mathcal{D}$ is a function that maps each variable $X_i \in \mathcal{X}$ to its finite discrete domain $\mathcal{D}(X_i)$, and $\mathcal{C}$ is a set of constraints. Each constraint $C \in \mathcal{C}$ is an ordered pair $\langle def, Y \rangle$, where $Y = \{Y_1, \ldots, Y_{|Y|}\} \subseteq \mathcal{X}$, denoted by $S(C)$, is a subset of all variables and def, denoted by $E_C(\cdot)$, is a function $\mathcal{D}(Y_1) \times \cdots \times \mathcal{D}(Y_{|Y|}) \to A$. A *solution* sol is an assignment of values to all variables, i.e., a function that maps each variable $X_i \in \mathcal{X}$ to its domain $\mathcal{D}(X_i)$. The total valuation (weight) of sol in P is defined as $W(sol) = \circledast_{C \in \mathcal{C}} E_C(sol|S(C))$. A solution sol is optimal iff, for any solution sol', $W(sol) \preceq W(sol')$ holds. A solution sol is called *α-qualified* with respect to a constraint C iff $E_C(sol|S(C)) \preceq \alpha$.

Similar to the case of SCSPs, a crisp CSP can be seen as a specialization of a VCSP. For example, a crisp CSP is a VCSP $P = \langle S = \langle E, \circledast, \prec, \top, \bot \rangle, \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ in which (a) $E = \{\top, \bot\}$ (thus $a \prec b$ iff $a = \bot$ and $b = \top$, and $\circledast$ can be any operator that complies the definition of a valuation structure), (b) in each constraint $C = \langle def, Y \rangle$, def maps forbidden assignments of values to $\top$ and allowed assignments of values to $\bot$, and (c) a solution to the crisp CSP is a solution sol such that $W(sol) = \bot$.

The Best-$\odot$ Message Passing Algorithm

Message passing algorithms solve various problems, such as constraint optimization and probabilistic marginalization, by passing local information between variables (Mézard and Montanari 2009). A message passing algorithm first builds a factor graph that reflects the interactions between variables and constraints (factors), then updates the messages between constraints and the variables participating in them according to given local update rules, and finally extracts a solution from these messages by inspecting local messages coming to each individual variable.

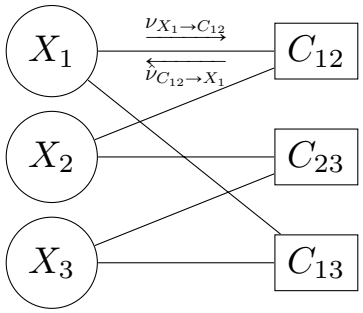


Figure 1: Illustrates the factor graph of an SCSP/VCSP with 3 variables $\{X_1, X_2, X_3\}$ and 3 constraints $\{C_{12}, C_{23}, C_{13}\}$. Here, $X_1, X_2 \in S(C_{12})$, $X_2, X_3 \in S(C_{23})$, and $X_1, X_3 \in S(C_{13})$. The circles represent variable vertices, and the squares represent constraint vertices. $\nu_{X_1 \to C_{12}}$ and $\hat{\nu}_{C_{12} \to X_1}$ are the messages from X_1 to C_{12} and from C_{12} to X_1, respectively. Such a pair of messages annotates each edge (even though not all of them are shown).

In this section, we introduce a message passing algorithm for solving SCSPs and VCSPs. It works as follows.

1. Construct an undirected bipartite graph G_f (factor graph), where each variable is represented by a vertex (variable vertex) in the first partition and each constraint is represented by a vertex (constraint vertex) in the second partition. (For convenience of exposition, we use "variable vertices" and "constraint vertices" interchangeably with "variables" and "constraints", respectively.) Connect X_i and C_j with an edge $\overline{X_i C_j}$ iff $X_i \in S(C_j)$. Figure 1 illustrates a factor graph.
2. Send messages in both directions along each edge. Messages $\nu_{X_i \to C_j}$ and $\hat{\nu}_{C_j \to X_i}$ are sent along edge $\overline{X_i C_j}$ from X_i to C_j and from C_j to X_i, respectively. Both messages are vectors of size $|\mathcal{D}(X_i)|$. Formally,
$$\nu_{X_i \to C_j} = \langle \nu_{X_i \to C_j}(X_i = x) \mid x \in \mathcal{D}(X_i) \rangle \quad (1)$$
$$\hat{\nu}_{C_j \to X_i} = \langle \hat{\nu}_{C_j \to X_i}(X_i = x) \mid x \in \mathcal{D}(X_i) \rangle. \quad (2)$$
Here, $\nu_{X_i \to C_j}(X_i = x)$ and $\hat{\nu}_{C_j \to X_i}(X_i = x)$ are called "components $X_i = x$" of $\nu_{X_i \to C_j}$ and $\hat{\nu}_{C_j \to X_i}$, respectively. Figure 1 illustrates the messages.
3. Initialize all messages to $\mathbf{1}$ for an SCSP and $\bot$ for a VCSP, and then perform update operations on them (i.e., update messages) iteratively according to
$$\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x) = \underset{a \in \mathcal{A}(\partial C_j \setminus \{X_i\})}{\mathrm{best}} \Big[E_{C_j}(a \cup \{X_i = x\}) \odot \big[\bigodot_{X_k \in \partial C_j \setminus \{X_i\}} \nu^{(t-1)}_{X_k \to C_j}(a|\{X_k\}) \big] \Big] \odot \hat{c}^{(t)}_{C_j \to X_i} \quad (3)$$
and
$$\nu^{(t)}_{X_i \to C_j}(X_i = x) = \left[\bigodot_{C_k \in \partial X_i \setminus \{C_j\}} \hat{\nu}^{(t-1)}_{C_k \to X_i}(X_i = x) \right] \odot c^{(t)}_{X_i \to C_j} \quad (4)$$
for all $X_i \in \mathcal{X}$, $C_j \in \mathcal{C}$, and $x \in \mathcal{D}(X_i)$, where

- ∂X_i and ∂C_j are the sets of adjacent vertices of X_i and C_j in G_f, respectively,
- $\mathcal{A}(\partial C_j \backslash \{X_i\})$ is the set of all assignments of values to variables in $\partial C_j \backslash \{X_i\}$ and $\mathcal{A}(\varnothing) = \{\varnothing\}$,
- superscript (t) indicates the update operation iteration index,
- $\odot$ is $\times$ for an SCSP and $\circledast$ for a VCSP,
- $\bigodot$ over an empty set yields $\mathbf{1}$ for an SCSP and $\bot$ for a VCSP, and
- $c^{(t)}_{X_i \to C_j}$ and $\hat{c}^{(t)}_{C_j \to X_i}$ are normalization factors that prevent messages from blowing up and can be set in different ways depending on the context (while they can always be set to $\mathbf{1}$ for an SCSP and thus are not required in this case).

Repeat this step until convergence, i.e., $\nu^{(t)}_{X_i \to C_j}(X_i = x) = \nu^{(t-1)}_{X_i \to C_j}(X_i = x)$ and $\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x) = \hat{\nu}^{(t-1)}_{C_j \to X_i}(X_i = x)$ hold for all $X_i \in \mathcal{X}$, $C_j \in \mathcal{C}$, and $x \in \mathcal{D}(X_i)$.

4. A set of values of all messages is called a *fixed point* iff it satisfies Eqs. (3) and (4) with $(t-1)$ set to (t) for all $X_i \in \mathcal{X}$, $C_j \in \mathcal{C}$, and $x_i \in \mathcal{D}(X_i)$. Convergence in Step 3 always leads to a fixed point, and all messages at such a fixed point are denoted by the superscript (∞). A final assignment of values to all variables in $\mathcal{X}$ can then be found by computing

$$E_{X_i}(X_i = x_i) = \bigodot_{C_j \in \partial X_i} \hat{\nu}^{(\infty)}_{C_j \to X_i}(X_i = x_i) \tag{5}$$

for all $X_i \in \mathcal{X}$ and $x_i \in \mathcal{D}(X_i)$. By selecting the value of x_i that leads to a best value of $E_{X_i}(X_i = x_i)$, we obtain the final assignment of values to all variables in $\mathcal{X}$.

The message update rules Eqs. (3) and (4) can be intuitively understood as follows. Each message from a variable vertex X_i to a constraint vertex C_j is updated by using $\odot$ over all of X_i's incoming messages from its other adjacent vertices. Each message from a constraint vertex C_j to a variable vertex X_i is updated by finding the best of the constraint function $E_{C_j} \odot$ all C_j's incoming messages from its other neighboring vertices. The reason that normalization constants are required in the message passing rules for VCSPs is as follows: During the iterative update procedure, the values of the messages may keep increasing or decreasing. In theory, this may cause convergence to never happen. In practice, this causes overflow or underflow. Normalization can effectively stop this trend. This, however, is not required for SCSP—as we show later in Theorem 1, the BOMP algorithm always converges without normalization with a given order for updating messages.

Since the message update rules of Eqs. (3) and (4) involve only two operators, namely best and $\odot$, corresponding to the operators in an SCSP/VCSP, we name this message passing algorithm the *best-$\odot$ message passing* (BOMP) algorithm (where "O" stands for $\odot$, called "O dot"). The BOMP algorithm is a generalization of other standard message passing algorithms, such as the min-sum and max-product message passing algorithms (Mézard and Montanari 2009). Similar to them, the BOMP algorithm neither specifies the order of the message updates in Step 3, nor provides any guarantee for the correctness or convergence of the final assignment in general.

The BOMP Algorithm on an SCSP

In this section, we study the BOMP algorithm on an SCSP. We show that, unlike other message passing algorithms, it always converges in polynomial time and enforces AC.

Convergence

In this subsection, we formally prove that the BOMP algorithm always converges in polynomial time for an SCSP, even though convergence is not guaranteed for other well-known message passing algorithms such as the min-sum and max-product message passing algorithms.

We now rewrite Eqs. (3) and (4) specialized for an SCSP as

$$\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x) = \mathop{+}_{a \in \mathcal{A}(\partial C_j \backslash \{X_i\})} \Big[E_{C_j}(a \cup \{X_i = x\}) \times \big[\mathop{\times}_{X_k \in \partial C_j \backslash \{X_i\}} \nu^{(t-1)}_{X_k \to C_j}(a|\{X_k\}) \big] \Big] \tag{6}$$

$$\nu^{(t)}_{X_i \to C_j}(X_i = x) = \left[\mathop{\times}_{C_k \in \partial X_i \backslash \{C_j\}} \hat{\nu}^{(t-1)}_{C_k \to X_i}(X_i = x) \right], \tag{7}$$

where the normalization constants are removed since they can always be set to $\mathbf{1}$ for an SCSP.

Lemma 1 (reflexivity, antisymmetry, transitivity)**.** *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, the reflexivity and antisymmetry properties of $\leqslant_S$ hold, i.e., $\forall a \in A : a \leqslant_S a$ and $\forall a, b \in A : a \leqslant_S b \wedge b \leqslant_S a \Leftrightarrow a = b$. The transitivity properties of $\leqslant_S$ and $<_S$ hold, i.e., $\forall a, b, c \in A : a \leqslant_S b \wedge b \leqslant_S c \Rightarrow a \leqslant_S c$ and $\forall a, b, c \in A : a <_S b \wedge b <_S c \Rightarrow a <_S c$.*

Proof. Reflexivity: $a + a = a \Rightarrow a \leqslant_S a$.

Antisymmetry: $a \leqslant_S b \wedge b \leqslant_S a \Leftrightarrow a + b = b \wedge a + b = a \Leftrightarrow a = b$.

Transitivity ($\leqslant_S$): $c = b + c = (a + b) + c = a + (b + c) = a + c \Leftrightarrow a \leqslant_S c$.

Transitivity ($<_S$): $\left.\begin{array}{ll} a <_S b & \Leftrightarrow a \leqslant_S b \wedge a \neq b \\ b <_S c & \Leftrightarrow b \leqslant_S c \wedge b \neq c \end{array}\right\} \Rightarrow a \leqslant_S c$. $a \neq c$ since, if $a = c$, then $b <_S c \Rightarrow b <_S a$, which contradicts with $a <_S b$. Therefore, $a <_S c$. □

Lemma 2. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, we have $\forall a, b, c \in A : a \leqslant_S b \Rightarrow a \times c \leqslant_S b \times c$.*

Proof. $a \times c + b \times c = (a + b) \times c = b \times c$. □

Lemma 3. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, for any $a, a', b, b', c, c' \in A$, we have $c \not\geqslant_S c' \Rightarrow a \not\geqslant_S a' \vee b \not\geqslant_S b'$ if either (a) $c = a \times b$ and $c' = a' \times b'$ or (b) $c = a + b$ and $c' = a' + b'$ hold.*

Proof by contradiction. First, $\forall x, y, z \in A : x \geqslant_S y \Rightarrow x + z \geqslant_S y + z$ holds, since $(x + z) + (y + z) = (x + y) + z = x + z$.

Now assume $a \geqslant_S a' \wedge b \geqslant_S b'$. Then, with Lemma 2, both (a) $c = a \times b \geqslant_S a' \times b \geqslant_S a' \times b' = c'$, and (b) $c = a + b \geqslant_S a' + b \geqslant_S a' + b' = c'$ hold, which contradicts $c \not\geqslant_S c'$. Therefore, this lemma holds. □

Lemma 4. *When applying the BOMP algorithm to an SCSP $P = \langle S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle, \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$, any component of any message is changed to a worse value in any update operation.*

Proof by induction. This lemma holds trivially for the first update operation, since all components of all messages are initialized to $\mathbf{1}$ (since $\forall a \in A : a \leqslant_S \mathbf{1}$).

Assume that the lemma holds for the first t update operations. Consider the $(t+1)^{\text{th}}$ update operation and a component of a message from a constraint vertex to a variable vertex such that $\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x) \not\geqslant_S \hat{\nu}^{(t+1)}_{C_j \to X_i}(X_i = x)$. Since $E_{C_j}(a \cup \{X_i = x\})$ does not change, from Eq. (6) and Lemma 3, $\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x) \not\geqslant_S \hat{\nu}^{(t+1)}_{C_j \to X_i}(X_i = x)$ implies that there must exist an $X_k \in \partial C_j \backslash \{X_i\}$, a $t' < t$, and an $x' \in \mathcal{D}(X_k)$ such that $\nu^{(t')}_{X_k \to C_j}(X_k = x') \not\geqslant_S \nu^{(t'+1)}_{X_k \to C_j}(X_k = x')$, which contradicts the induction assumption. From Eq. (7) and Lemma 3, a similar contradiction occurs for messages from variable vertices to constraint vertices. Thus, this lemma continues to hold for the first $(t+1)$ update operations and is therefore generally true. □

Theorem 1. *For an SCSP, there exists an order of message update operations such that the running time of the BOMP algorithm is polynomially bounded.*

Proof. Let the BOMP algorithm update messages in a sweeping order, i.e., messages are updated in rounds, in each of which both messages along each edge are updated once in an arbitrary order. From Lemma 4, the BOMP algorithm terminates after $\mathcal{O}(|A| \cdot \max_{X_i \in \mathcal{X}} |\mathcal{D}(X_i)| \cdot \max_{C_j \in \mathcal{C}} |S(C_j)| \cdot |\mathcal{C}|)$ rounds. This upper bound represents the product of the total number of components of all messages and the number of times that any of these components can change. □

Relationship between BOMP and AC

In this subsection, we study the relationship between the BOMP algorithm and AC in binary SCSPs. Throughout this subsection, we assume that all constraints are unary or binary, i.e., for any constraint C, $|S(C)| \in \{1, 2\}$. Without loss of generality, we assume that there always exists exactly 1 unary constraint on each individual variable X_i (denoted by C_i) and at most 1 binary constraint on each pair of distinct variables X_i, X_j (denoted by C_{ij}). Otherwise, overlapping constraints on the same variables can be collapsed into one single constraint by using the $\times$ operator.

The relationship between SCSPs and the BOMP algorithm can be established by using the concept of *message passing unary constraints*. This relationship does not rely on the final solution extracted from Step 4 of the BOMP algorithm.

Definition 8. For a binary SCSP $P = \langle S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle, \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$, upon convergence of the BOMP algorithm to a fixed point F, we define the *message passing unary constraint* C_i^F of variable X_i at F to be

$$E_{C_i^F}(\{X_i = x_i\}) = \bigtimes_{C \in \partial X_i} \hat{\nu}^{(\infty)}_{C \to X_i}(X_i = x_i), \tag{8}$$

or, from Eq. (7), equivalently,

$$\forall C \in \partial X_i : E_{C_i^F}(\{X_i = x_i\}) = \nu^{(\infty)}_{X_i \to C}(X_i = x_i) \times \hat{\nu}^{(\infty)}_{C \to X_i}(X_i = x_i). \tag{9}$$

The set of all message passing unary constraints is denoted by $\mathcal{C}^F$. $\mathcal{D}^F_\alpha$ is a function that maps each variable X_i to its *message passing domain* $\mathcal{D}^F_\alpha(X_i)$ with respect to a parameter $\alpha \in A$:

$$\mathcal{D}^F_\alpha(X_i) = \left\{ x_i \in \mathcal{D}(X_i) \mid E_{C_i^F}(\{X_i = x_i\}) \geqslant_S \alpha \right\}. \tag{10}$$

Lemma 5. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where $\times$ is idempotent, we have $\forall a, b \in A : a \leqslant_S b \Leftrightarrow a \leqslant_S a \times b$.*

Proof. ($\Rightarrow$): $a \leqslant_S b \Rightarrow a \times a \leqslant_S a \times b \Rightarrow a \leqslant_S a \times b$.

($\Leftarrow$): $a \leqslant_S a \times b \Leftrightarrow a + a \times b = a \times b \Leftrightarrow a \times (\mathbf{1} + b) = a \times b \Leftrightarrow a = a \times b \Rightarrow a + b = a \times b + b = (a + \mathbf{1}) \times b = b \Rightarrow a \leqslant_S b$. □

Lemma 6. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$, we have $\forall a, b \in A : a \times b \leqslant_S a$.*

Proof. $a \times b + a = a \times (b + \mathbf{1}) = a \times \mathbf{1} = a \Rightarrow a \times b \leqslant_S a$. □

Lemma 7. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where $\times$ is idempotent, we have $\forall a, b \in A : a \leqslant_S b \Leftrightarrow a = a \times b$. More generally, we have $\forall a, b_1, \ldots, b_k \in A : a \leqslant_S b_1 \wedge \cdots \wedge a \leqslant_S b_k \Leftrightarrow a = a \times b_1 \times \cdots \times b_k$.*

Proof. ($\Rightarrow$): $a \leqslant_S b \Leftrightarrow a + b = b \Rightarrow a \times (a + b) = a \times b \Leftrightarrow a \times a + a \times b = a \times b \Leftrightarrow a + a \times b = a \times b \Leftrightarrow a \leqslant_S a \times b$. Since $a \times b \leqslant_S a$ (from Lemma 6), we have $a = a \times b$.

($\Leftarrow$): From Lemma 6, $a \times b \leqslant_S b$. Therefore, $a = a \times b \leqslant_S b$.

By recursively applying the first half of this lemma, we have the second half of this lemma. □

In the case of A being totally ordered, intuitively, Lemma 7 states that the product of multiple variables equals the worst one of them.

Lemma 8. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where $\times$ is idempotent and A is totally ordered, we have $\forall a, b_1, \ldots, b_k \in A : a \leqslant_S b_1 \times \cdots \times b_k \Leftrightarrow a \leqslant_S b_1 \wedge \cdots \wedge a \leqslant_S b_k$.*

Proof. Since A is totally ordered, we can assume $b \leqslant_S \cdots \leqslant_S b_k$ without loss of generality. Then, from Lemma 7, $a \leqslant_S b_1 \times \cdots \times b_k \Leftrightarrow a = a \times b_1 \times \cdots \times b_k \Leftrightarrow a \leqslant_S b_1 \wedge \cdots \wedge a \leqslant_S b_k$. □

Lemma 9. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where $\times$ is idempotent and A is totally ordered, we have $\forall a, b, c \in A : a \times b \leqslant_S c \Leftrightarrow a \leqslant_S c \vee b \leqslant_S c$ and $\forall a, b, c \in A : a \times b <_S c \Leftrightarrow a <_S c \vee b <_S c$.*

Proof. ($\Rightarrow$): Since A is totally ordered, we can assume $a \leqslant_S b$ without loss of generality. We prove this by contradiction. We assume, for a proof by contradiction, that $a >_S c \wedge b >_S c$. Then, from Lemma 7, $a \times b = a >_S c$, which contradicts the left-hand side.

($\Leftarrow$): $a \times b = a \leqslant_S c$.

The second part of this lemma can be proved similarly. □

Lemma 10. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where $\times$ is idempotent and A is totally ordered, we have $\forall a, b, c \in A : a \times b \leqslant_S c \wedge b >_S c \times a \Rightarrow a \leqslant_S c$.*

Proof. From Lemma 9, $a \leqslant_S c \vee b \leqslant_S c$ and $c <_S b \vee a <_S b$. $b \leqslant_S c$ and $c <_S b$ cannot hold at the same time, since they together imply $b <_S b$. Therefore, if $b \leqslant_S c$ holds, then $a <_S b$ must hold, which implies $a \leqslant_S c$; if $b \leqslant_S c$ does not hold, then $c <_S b \vee a <_S b$ and $a \leqslant_S c$ hold. □

Lemma 11. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$, we have $\forall a, b, c \in A : a \leqslant_S b \Rightarrow a \leqslant_S b + c$.*

Proof. $a \leqslant_S b \Leftrightarrow a + b = b \Rightarrow a + b + c = b + c \Leftrightarrow a \leqslant_S b + c$. □

Lemma 12. *In a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where A is totally ordered, we have $\forall a, b, c \in A : a + b \leqslant_S c \Leftrightarrow a \leqslant_S c \wedge b \leqslant_S c$ and $\forall a, b, c \in A : a + b <_S c \Leftrightarrow a <_S c \wedge b <_S c$.*

Proof. ($\Rightarrow$): Since A is totally ordered, we can assume $a \leqslant_S b$ without loss of generality. With this and $a + b \leqslant_S c$, we have $b + c = a + b + c = c \Rightarrow b \leqslant_S c$. In addition, we have $a \leqslant_S b \wedge b \leqslant_S c \Rightarrow a \leqslant_S c$.

($\Leftarrow$): $a \leqslant_S c \wedge b \leqslant_S c \Leftrightarrow a + c = c \wedge b + c = c \Rightarrow a + b + c = c \Leftrightarrow a + b \leqslant_S c$.

We again assume $a \leqslant_S b$. Then $a + b <_S c \Leftrightarrow a + b \neq c \wedge a + b \leqslant_S c \Leftrightarrow b \neq c \wedge a \leqslant_S c \wedge b \leqslant_S c \Leftrightarrow a \leqslant_S c \wedge b <_S c \Leftrightarrow a \leqslant_S b <_S c$. □

We prove the following two theorems for any SCSP with a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ where $\times$ is idempotent and A is totally ordered. These SCSPs cover many important CSP generalizations including fuzzy CSPs (Bistarelli et al. 1999). We also hope that this proof will turn out to be inspiring for proving properties of the BOMP algorithm for more general cases.

Theorem 2. *For any SCSP $P = \langle S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle, \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$ where $\times$ is idempotent and A is totally ordered, at any fixed point F to which the BOMP algorithm converges on P, the SCSP $P' = \langle S, \mathcal{X}, \mathcal{D}, \mathcal{C}^b \cup \mathcal{C}^F\rangle$ is arc-consistent, where $\mathcal{C}^b$ is the set of all binary constraints in $\mathcal{C}$.*

Proof. Proving this theorem is equivalent to proving that, for any two variables X_i and X_j,

$$\begin{aligned}&\forall x_i \in \mathcal{D}(X_i) : E_{C_i^F}(\{X_i = x_i\}) \leqslant_S \\ &\quad \underset{x_j \in \mathcal{D}(X_j)}{+} \left[E_{C_i^F}(\{X_i = x_i\}) \times E_{C_j^F}(\{X_j = x_j\}) \times \right. \\ &\quad \left. E_{C_{ij}}(\{X_i = x_i, X_j = x_j\})\right] = \\ &\quad E_{C_i^F}(\{X_i = x_i\}) \times \\ &\quad \underset{x_j \in \mathcal{D}(X_j)}{+} \left[E_{C_j^F}(\{X_j = x_j\}) \times E_{C_{ij}}(\{X_i = x_i, X_j = x_j\})\right]\end{aligned} \tag{11}$$

holds. From Lemma 5, to prove this, it is equivalent to prove

$$\begin{aligned}&\forall x_i \in \mathcal{D}(X_i) : E_{C_i^F}(\{X_i = x_i\}) \leqslant_S \\ &\quad \underset{x_j \in \mathcal{D}(X_j)}{+} \left[E_{C_j^F}(\{X_j = x_j\}) \times E_{C_{ij}}(\{X_i = x_i, X_j = x_j\})\right].\end{aligned} \tag{12}$$

To prove this, from Lemma 11, it is sufficient to prove

$$\begin{aligned}&\forall x_i \in \mathcal{D}(X_i) : \exists x_j \in \mathcal{D}(X_j) : E_{C_i^F}(\{X_i = x_i\}) \leqslant_S \\ &\quad E_{C_j^F}(\{X_j = x_j\}) \times E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}).\end{aligned} \tag{13}$$

We prove this by contradiction. We assume that there exists a fixed point F' such that

$$\begin{aligned}&\exists x_i \in \mathcal{D}(X_i) : \forall x_j \in \mathcal{D}(X_j) : E_{C_i^{F'}}(\{X_i = x_i\}) >_S \\ &\quad E_{C_j^{F'}}(\{X_j = x_j\}) \times E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}).\end{aligned} \tag{14}$$

We now consider such an x_i. We define $\mathcal{D}^{F'}(X_j)$ as

$$\mathcal{D}^{F'}(X_j) = \left\{x_j \in \mathcal{D}(X_j) \mid E_{C_j^{F'}}(\{X_j = x_j\}) = E_{C_j}(\{X_j = x_j\})\right\}. \tag{15}$$

We now prove

$$\begin{aligned}&\forall x_j \in \mathcal{D}(X_j) : E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \times E_{C_j^{F'}}(\{X_j = x_j\}) \\ &\quad \geqslant_S E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \times \nu^{(\infty)}_{X_j \to C_{ij}}(X_j = x_j)\end{aligned} \tag{16}$$

for two different cases, $x_j \in \mathcal{D}^{F'}(X_j)$ and $x_j \in \mathcal{D}(X_j) \backslash \mathcal{D}^{F'}(X_j)$.

- $x_j \in \mathcal{D}^{F'}(X_j)$: From Eqs. (6) and (7) and Lemma 6, we have

$$\begin{aligned}&\forall x_j \in \mathcal{D}^{F'}(X_j) : E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \times E_{C_j^{F'}}(\{X_j = x_j\}) \\ &\quad = E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \times \hat{\nu}^{(\infty)}_{C_j \to X_j}(X_j = x_j) \quad \text{(Eq. (6))} \\ &\quad \geqslant_S E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \times \nu^{(\infty)}_{X_j \to C_{ij}}(X_j = x_j).\end{aligned} \tag{17}$$

  This implies Eq. (16) for this case.

- $x_j \in \mathcal{D}(X_j) \backslash \mathcal{D}^{F'}(X_j)$: We have

$$\begin{aligned}&\forall x_j \in \mathcal{D}(X_j) \backslash \mathcal{D}^{F'}(X_j) : E_{C_j^{F'}}(\{X_j = x_j\}) \overset{\text{Eq. (15)}}{=} \\ &\quad E_{C_j}(\{X_j = x_j\}) \overset{\text{Eq. (6)}}{=} \hat{\nu}^{(\infty)}_{C_j \to X_j}(X_j = x_j).\end{aligned} \tag{18}$$

  In addition, from Eq. (8) and Lemma 7, we have

$$\begin{aligned}&\forall x_j \in \mathcal{D}(X_j) \backslash \mathcal{D}^{F'}(X_j) : \exists C \in \partial X_j : \\ &\quad \hat{\nu}^{(\infty)}_{C \to X_j}(X_j = x_j) = E_{C_j^{F'}}(\{X_j = x_j\}).\end{aligned} \tag{19}$$

  Therefore, we have

$$\begin{aligned}&\forall x_j \in \mathcal{D}(X_j) \backslash \mathcal{D}^{F'}(X_j) : \exists C \in \partial X_j \backslash \{C_j\} : \\ &\quad \hat{\nu}^{(\infty)}_{C \to X_j}(X_j = x_j) = E_{C_j^{F'}}(\{X_j = x_j\}).\end{aligned} \tag{20}$$

  For all such x_j's, it is sufficient to prove

$$\begin{aligned}&\forall x_j \in \mathcal{D}(X_j) \backslash D^{F'}(X_j) : E_{C_j^{F'}}(\{X_j = x_j\}) \geqslant_S \\ &\quad E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \times \nu^{(\infty)}_{X_j \to C_{ij}}(X_j = x_j),\end{aligned} \tag{21}$$

  since, from Lemma 2, it implies Eq. (16). We consider two subcases:

  – x_j satisfies $\exists C \in \partial X_j \backslash \{C_j, C_{ij}\} : \hat{\nu}^{(\infty)}_{C \to X_j}(X_j = x_j) = E_{C_j^{F'}}(\{X_j = x_j\})$. From Eq. (8) and Lemma 8, we have

$$\begin{aligned}&\forall x_j \in \mathcal{D}(X_j) \backslash \mathcal{D}^{F'}(X_j) : \forall C \in \partial X_j : \\ &\quad \hat{\nu}^{(\infty)}_{C \to X_j}(X_j = x_j) \geqslant_S E_{C_j^{F'}}(\{X_j = x_j\}).\end{aligned} \tag{22}$$

    Then, by applying Eq. (7) to $\nu^{(\infty)}_{X_j \to C_{ij}}(X_j = x_j)$ and, from Lemma 7, we have

$$\nu^{(\infty)}_{X_j \to C_{ij}}(X_j = x_j) = E_{C_j^{F'}}(\{X_j = x_j\}). \tag{23}$$

    From Lemma 6, this implies Eq. (21) for this subcase.

– x_j satisfies $\forall C \in \partial X_j \backslash \{C_j, C_{ij}\} : \hat{\nu}^{(\infty)}_{C \to X_j}(X_j = x_j) = E_{C_j^{F'}}(\{X_j = x_j\})$. From Eq. (20), this implies $\hat{\nu}^{(\infty)}_{C_{ij} \to X_j}(X_j = x_j) = E_{C_j^{F'}}(\{X_j = x_j\})$ (and thus $\hat{\nu}^{(\infty)}_{C_{ij} \to X_j}(X_j = x_j) \leqslant_S E_{C_j^{F'}}(\{X_j = x_j\})$). By applying Eq. (6) to $\hat{\nu}^{(\infty)}_{C_{ij} \to X_j}(X_j = x_j)$, and, from Lemma 12, we have

$$E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \times \nu^{(\infty)}_{X_i \to C_{ij}}(X_i = x_i) \leqslant_S E_{C_j^{F'}}(\{X_j = x_j\}). \tag{24}$$

Then we have

$$\nu^{(\infty)}_{X_i \to C_{ij}}(X_i = x_i) >_S E_{C_j^{F'}}(\{X_j = x_j\}) \times E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}), \tag{25}$$

because (a) if $\partial X_i = \{C_{ij}, C_i\}$, then

$$\begin{aligned}
&\nu^{(\infty)}_{X_i \to C_{ij}}(X_i = x_i) = \hat{\nu}^{(\infty)}_{C_i \to X_i}(X_i = x_i) \quad \text{(Eq. (7))} \\
&\geqslant_S E_{C_i^{F'}}(\{X_i = x_i\}) \quad \text{(Eq. (8) (with = replaced by } \leqslant_S \text{) and Lemma 8)} \\
&>_S E_{C_j^{F'}}(\{X_j = x_j\}) \times E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \quad \text{(Eq. (14)),}
\end{aligned}$$

or (b) otherwise, if we assume that Eq. (25) does not hold, since Eq. (9) (with = replaced by $\leqslant_S$) and Lemma 8 imply $E_{C_i^{F'}}(\{X_i = x_i\}) \leqslant_S \nu^{(\infty)}_{X_i \to C_{ij}}(X_i = x_i)$, we have $E_{C_i^{F'}}(\{X_i = x_i\}) \leqslant_S E_{C_j^{F'}}(\{X_j = x_j\}) \times E_{C_{ij}}(\{X_i = x_i, X_j = x_j\})$, which contradicts the assumption (Eq. (14)).
Equations (24) and (25) and Lemma 10 imply

$$E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \leqslant_S E_{C_j^{F'}}(\{X_j = x_j\}). \tag{26}$$

From Lemma 6, this implies Eq. (21) for this subcase.

By combining Eqs. (14) and (16), we have

$$\forall x_j \in \mathcal{D}(X_j) : E_{C_i^{F'}}(\{X_i = x_i\}) >_S E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) \times \nu^{(\infty)}_{X_j \to C_{ij}}(X_j = x_j). \tag{27}$$

By applying Eq. (6) (with = replaced by $\geqslant_S$) to $\hat{\nu}^{(\infty)}_{C_{ij} \to X_i}(X_i = x_i)$ and from Lemma 12, we have

$$E_{C_i^{F'}}(\{X_i = x_i\}) >_S \hat{\nu}^{(\infty)}_{C_{ij} \to X_i}(X_i = x_i), \tag{28}$$

which along with Lemma 6 contradicts Eq. (8). □

While Theorem 2 applies to SCSPs with only binary constraints, there exist algorithms that convert any SCSP constraints to binary SCSP constraints (Larrosa and Dechter 2000). Therefore, this theorem is still applicable to general SCSPs if their constraints are converted to binary constraints.

Theorem 3. *For any SCSP $P = \langle S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle, \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$ where $\times$ is idempotent and S is totally ordered, the SCSP $P'' = \langle S, \mathcal{X}, \mathcal{D}^F_\alpha, \mathcal{C}\rangle$ preserves all solutions with total weights better than α for any fixed point F to which the BOMP algorithm converges on P.*

Proof. We construct a crisp CSP $P' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}'\rangle$ from P, where each $C' \in \mathcal{C}'$ has a one-to-one correspondence with a constraint $C \in \mathcal{C}$ over the same variables. C' allows an assignment a of values to variables in $S(C')$ iff $E_C(a) \geqslant_S \alpha$. Alternatively, we define $E_{C'}(a) = M(E_C(a))$, where $M : A \to \{0, 1\}$ is

$$M(x) = \begin{cases} 0, & \text{if } x \geqslant_S \alpha \\ 1, & \text{otherwise.} \end{cases} \tag{29}$$

Here, 0 and 1 represent allowed and disallowed assignments of values to variables, respectively. When applying the BOMP algorithm on P', if $\times$ and $+$ are replaced by the $\max$ and $\min$ operators, respectively, the values of messages obtained in the intermediate steps (Eqs. (6) and (7)) are the same as applying M to the values of the corresponding messages in the BOMP algorithm on P. The BOMP algorithm is equivalent to the min-max message passing (MMMP) algorithm on P' (Xu, Kumar, and Koenig 2017a). D^F_α in P'' is also equivalent to the message passing domains defined in the context of the MMMP algorithm on P'. From Theorem 3 in (Xu, Kumar, and Koenig 2017a), after applying the BOMP algorithm on P', all solutions to P' are preserved. From Eq. (29), a solution to P' is valid iff it has a total weight better than α in P. Therefore, all solutions to P with total weights better than α are preserved in P''. □

The BOMP Algorithm on VCSPs

In this section, we study the BOMP algorithm on VCSPs.

We now rewrite Eqs. (3) and (4) specialized for a VCSP as

$$\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x) = \underset{a \in \mathcal{A}(\partial C_j \backslash \{X_i\})}{\text{best}} \Big[E_{C_j}(a \cup \{X_i = x\}) \circledast \Big[\circledast_{X_k \in \partial C_j \backslash \{X_i\}} \nu^{(t-1)}_{X_k \to C_j}(a[\{X_k\}]) \Big] \circledast \hat{c}^{(t)}_{C_j \to X_i} \tag{30}$$

$$\nu^{(t)}_{X_i \to C_j}(X_i = x) = \Big[\circledast_{C_k \in \partial X_i \backslash \{C_j\}} \hat{\nu}^{(t-1)}_{C_k \to X_i}(X_i = x) \Big] \circledast c^{(t)}_{X_i \to C_j}. \tag{31}$$

Here, in practice, the normalization constants are usually chosen to avoid issues such as overflow or underflow. For example, a weighted constraint satisfaction problem (WCSP) can be seen as a VCSP, in which E is a set of non-negative real numbers, $\circledast$ is the addition operator on real numbers, $\prec$ is the "less than" operator of real numbers, and $\top, \bot$ are $+\infty$ and 0, respectively. A value $a \in E$ is better if it is smaller. The normalization constants can be set such that the best component in the message at each iteration is zero.

VCSPs with an Idempotent $\circledast$

In general, the BOMP algorithm on VCSPs does not have many useful properties beyond those common to all message passing algorithms. However, for a VCSP with an idempotent $\circledast$, many properties of the BOMP algorithm on SCSPs carry over, since these VCSPs can be formulated as SCSPs.

Formally, a VCSP $P = \langle S = \langle E, \circledast, \prec, \top, \bot\rangle, \mathcal{X}, \mathcal{D}, \mathcal{C}\rangle$, where $\circledast$ is idempotent, can be rewritten as an SCSP $P' = \langle S' = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle, \mathcal{X}', \mathcal{D}', \mathcal{C}'\rangle$ as follows. $\mathcal{X} = \mathcal{X}'$, $\mathcal{D} = \mathcal{D}'$, $\mathcal{C}' = \mathcal{C}$, $E = A$, $\mathbf{1} = \bot$ and $\mathbf{0} = \top$. $\times$ is the same as $\circledast$. The total order on E defined by $\succeq$ should be the same as the total order on A defined by $\leqslant_S$, and $+$ is defined to comply with this requirement of $\leqslant_S$. Under this transformation, we have the following theorems.

Table 1: "Total", "Converged", "Rate", and "Timeout" refer to the total number of benchmark instances, the number of converged benchmark instances, the fraction of converged benchmark instances, and the number of benchmark instances that did not finish 10,000 iterations within 5 minutes, respectively.

Instance set	Total	Converged	Rate	Timeout
UAI (MPMP)	173	140	80.92%	20
UAI (MSMP)	173	139	80.34%	19
toulbar2 (MSMP)	2220	1748	78.74%	418

Theorem 4. *Using the transformation procedure above, a solution sol for a VCSP $P = \langle S = \langle E, \circledast, \prec, \top, \bot \rangle, \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is optimal iff it is optimal for the SCSP $P' = \langle S' = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle, \mathcal{X}', \mathcal{D}', \mathcal{C}' \rangle$.*

Proof. We are required to prove that

$$\begin{aligned} \forall sol \in \mathcal{A}(\mathcal{X}) : ((\forall sol' \in \mathcal{A}(\mathcal{X}) : W(sol) \preceq W(sol')) \\ \Leftrightarrow (\forall sol' \in \mathcal{A}(\mathcal{X}) : W'(sol) \succcurlyeq_S W'(sol'))) \end{aligned} \tag{32}$$

holds, where $W(sol)$ and $W'(sol)$ are the valuations of sol in P and P', respectively.

In the transformation, $W(\cdot)$ and $W'(\cdot)$ are equivalent (since $\times$ is the same as $\circledast$), and $\preceq$ is the same as $\succcurlyeq_S$. Therefore, the equation above holds and implies that this theorem holds as well. □

Theorem 5. *For a VCSP with an idempotent $\circledast$, there exists an order of message update operations such that the running time of the BOMP algorithm is polynomially bounded (provided that the normalization constants are $\bot$).*

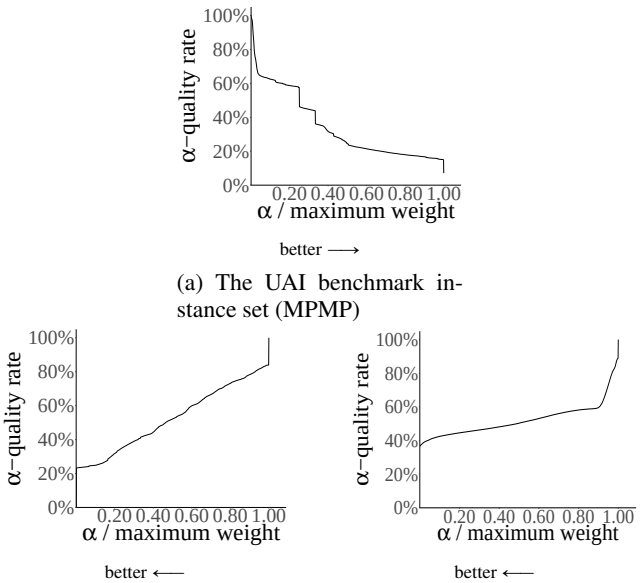
Proof. This theorem follows from Theorems 1 and 4. □

VCSPs in General

If $\circledast$ is not idempotent, then the convergence of the BOMP algorithm is not guaranteed. Even upon convergence, the relationship between the BOMP algorithm and AC is unclear, despite some known relationships between message passing algorithms on specialized instances of VCSP and AC (Dechter and Mateescu 2003). In this section, we therefore empirically study the convergence of the BOMP algorithm and the α-quality of solutions produced by it.

In our experiments, we used two benchmark instance sets: the PR benchmark instances from the UAI 2014 Inference Competition[1] (which we refer to as the UAI benchmark instance set) and the WCSP benchmark instances from (Hurley et al. 2016)[2] (which we refer to as the toulbar2 benchmark instance set), which includes benchmark instances from the Probabilistic Inference Challenge 2011, the Computer Vision and Pattern Recognition OpenGM2 benchmark, the Weighted Partial MaxSAT Evaluation 2013, the MaxCSP 2008 Competition, the MiniZinc Challenge 2012 & 2013, and the CFLib (a library of cost function networks).

For the UAI benchmark instances, we used the max-product message passing (MPMP) algorithm (Koller and Friedman 2009), i.e., the BOMP algorithm for VCSPs in which E is the set of real numbers between 0 and 1, $\bot$ is 1, $\top$ is 0, and $\circledast$ is the multiplicative operator on real numbers. A value $a \in E$ is better if it is closer to 0 and worse if it is closer to 1. The normalization constants in Eqs. (30) and (31) are chosen so that the sum of all components of each updated message is always 1. For the UAI benchmark instances, we also used the min-sum message passing (MSMP) algorithm, the message passing algorithm that solves WCSPs (Xu, Kumar, and Koenig 2017b), by using the negative logarithms of the weights in the factor tables after normalizing them. In this algorithm, the normalization constants in Eqs. (30) and (31) are chosen so that the smallest (best) component of each updated message is always 0. For the toulbar2 benchmark instances, we used the MSMP algorithm. We initialized the messages to 1's and 0's in the MPMP and MSMP algorithms, respectively. For each benchmark instance, for each constraint C in the order specified by the input file, we first updated all messages to C and then updated all messages from C. Convergence was declared iff all messages had differences less than 10^{-4} between two consecutive iterations. If the runs did not converge after 10,000 iterations (of message update operations), they were terminated. In addition to this termination condition, we also enforced a 5-minute running time limit.[3] Table 1 shows the convergence results.



(a) The UAI benchmark instance set (MPMP)

(b) The UAI benchmark instance set (MSMP)

(c) The toulbar2 benchmark instance set (MSMP)

Figure 2: The α-quality rates of the solutions sol produced by the BOMP algorithm versus α / maximum weight for the two benchmark instance sets. The x-axes are normalized by dividing α by the maximum weight among all constraints, $\max_C E_C(sol|S(C))$, in each benchmark instance.

[1] http://www.hlt.utdallas.edu/~vgogate/uai14-competition/

[2] http://genoweb.toulouse.inra.fr/~degivry/evalgm/

[3] The BOMP algorithm was implemented in C++ and compiled by clang(800.0.42.1)+LLVM(8.0.0), and was run on a MacBook with an Intel Core i5 processor (3MB Cache, 2.7 GHz) and 8GB RAM.

To elaborate on the experimental results, we use the concept of the α-*quality rate* of a solution sol, i.e., the percentage of constraints with respect to which sol is α-qualified. For each benchmark instance, we extracted a final solution sol using Step 4 of the BOMP algorithm. We then checked whether sol is α-qualified with respect to each constraint. Figure 2 shows the average α-quality rate over all benchmark instances versus α in each benchmark instance set. As expected, the α-quality rate decreases as α becomes better (as defined by $\succeq$). However, the nature of these curves depends on the formulation of the problem instances and the BOMP algorithm that works with it. For example, even within the same benchmark instance set (the UAI benchmark instance set in our experiments), the α-qualities of the solutions produced by the MPMP and MSMP algorithms exhibit different curves. This indicates that the BOMP algorithm is sensitive to the choice of the best and $\circledast$ operators as well as the normalization constants. We also note that the MSMP algorithm has a tendency to produce sharp turns, markedly for the points of $\alpha \approx 0.2$ in the UAI benchmark instance set and $\alpha \approx 0.9$ in the toulbar2 benchmark instance set. Our empirical observations indicate the need for a deeper understanding of the connections between message passing and different formulations of combinatorial problems.

Conclusions and Future Work

In this paper, we developed the BOMP algorithm for SCSPs and VCSPs. Our study facilitates the understanding and applicability of message passing techniques—popularly used for solving large-scale optimization problems—to expressive frameworks such as SCSPs and VCSPs. We proved the convergence of the BOMP algorithm on SCSPs as well as VCSPs with an idempotent $\circledast$. We established a theoretical connection between AC and the BOMP algorithm for SCSPs with an idempotent $\times$ and a totally ordered A upon convergence. We also empirically studied the α-quality of the solutions produced by the BOMP algorithm for general VCSPs. Future work includes developing deeper theoretical results that relate generalized message passing algorithms to higher levels of local consistency in SCSPs and VCSPs. In the same direction, we intend to bring search techniques used in the probabilistic reasoning and constraint reasoning communities closer to each other.

Acknowledgment The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1724392, 1409987, and 1319966.

References

Bistarelli, S.; Montanari, U.; Rossi, F.; Schiex, T.; Verfaillie, G.; and Fargier, H. 1999. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints* 4(3):199–240.

Dechter, R., and Mateescu, R. 2003. A simple insight into iterative belief propagation's success. In *the Annual Conference on Uncertainty in Artificial Intelligence*, 175–183.

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

Fioretto, F.; Xu, H.; Koenig, S.; and Kumar, T. K. S. 2018. Constraint composite graph-based lifted message passing for distributed constraint optimization problems. In *the International Symposium on Artificial Intelligence and Mathematics*.

Freuder, E. C. 1982. A sufficient condition for backtrack-free search. *Journal of the ACM* 29(1):24–32.

Hurley, B.; O'Sullivan, B.; Allouche, D.; Katsirelos, G.; Schiex, T.; Zytnicki, M.; and de Givry, S. 2016. Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints* 21(3):413–434.

Jeavons, P. G., and Cooper, M. C. 1995. Tractable constraints on ordered domains. *Artificial Intelligence* 79(2):327–339.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kolmogorov, V. 2005. Primal-dual algorithm for convex Markov random fields. Technical Report MSR-TR-2005-117, Microsoft Research.

Kolmogorov, V. 2006. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(10):1568–1583.

Larrosa, J., and Dechter, R. 2000. On the dual representation of non-binary semiring-based CSPs. In *Workshop 1 (Soft Constraints) of the International Conference on Principles and Practice of Constraint Programming*.

Larrosa, J., and Schiex, T. 2003. In the quest of the best form of local consistency for weighted CSP. In *the International Joint Conference on Artificial Intelligence*, 239–244.

Larrosa, J., and Schiex, T. 2004. Solving weighted CSP by maintaining arc consistency. *Artificial Intelligence* 159(1):1–26.

Mézard, M., and Montanari, A. 2009. *Information, Physics, and Computation*. Oxford University Press.

Mézard, M., and Zecchina, R. 2002. Random k-satisfiability problem: From an analytic solution to an efficient algorithm. *Physical Review E* 66(5):056126.

Nakajima, M.; Xu, H.; Koenig, S.; and Kumar, T. K. S. 2018. Towards understanding the min-sum message passing algorithm for the minimum weighted vertex cover problem: An analytical approach. In *the International Symposium on Artificial Intelligence and Mathematics*.

Weigt, M., and Zhou, H. 2006. Message passing for vertex covers. *Physical Review E* 74(4):046110.

Werner, T. 2015. Marginal consistency: Upper-bounding partition functions over commutative semirings. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37(7):1455–1468.

Xu, H.; Sun, K.; Koenig, S.; and Kumar, T. K. S. 2018. A warning propagation-based linear-time-and-space algorithm for the minimum vertex cover problem on giant graphs. In *the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 567–684.

Xu, H.; Kumar, T. K. S.; and Koenig, S. 2017a. Min-max message passing and local consistency in constraint networks. In *the Australasian Joint Conference on Artificial Intelligence*, 340–352.

Xu, H.; Kumar, T. K. S.; and Koenig, S. 2017b. The Nemhauser-Trotter reduction and lifted message passing for the weighted CSP. In *the International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming*, 387–402.

Zytnicki, M.; Gaspin, C.; and Schiex, T. 2008. DARN! A weighted constraint solver for RNA motif localization. *Constraints* 13(1):91–109.