Min-Max Message Passing and Local Consistency in Constraint Networks

Hong Xu, T. K. Satish Kumar, and Sven Koenig

University of Southern California, Los Angeles, CA 90089, USA
hongx@usc.edu tkskwork@gmail.com skoenig@usc.edu

Abstract. In this paper, we uncover some relationships between local consistency in constraint networks and message passing akin to belief propagation in probabilistic reasoning. We develop a new message passing algorithm, called the min-max message passing (MMMP) algorithm, for unifying the different notions of local consistency in constraint networks. In particular, we study its connection to arc consistency (AC) and path consistency. We show that AC-3 can be expressed more intuitively in the framework of message passing. We also show that the MMMP algorithm can be modified to enforce path consistency.

Keywords: Message Passing · Constraint Network · Local Consistency

1 Introduction

A *constraint network* (CN)—i.e., a *constraint satisfaction problem* instance—can be defined by a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ is the set of variables; $\mathcal{D}$, the *domain* of the CN, is a function that maps a variable to its discrete domain $\mathcal{D}(X_i)$; and $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ is the 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 is to solve the CN, i.e., find an assignment of values to all variables in $\mathcal{X}$ such that all constraints are satisfied. A constraint is satisfied iff it allows the assignment. CNs have been used to solve real-world combinatorial problems, such as map coloring and scheduling [7].

Local consistency of CNs is a class of properties over subsets of variables. A CN is said to be K-consistent iff, for any subset of $(K-1)$ variables, any consistent assignment of values to them (i.e., no constraints between them are violated) 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 is enforced by procedures that make implicit constraints explicit in a CN. Node consistency—i.e., 1-consistency—is the property that each value in the domain of a variable satisfies the unary constraint on it. Arc consistency (AC)—i.e., 2-consistency—is the property that each value in the domain of a variable has a consistent extension to any other variable. Path consistency (PC)—i.e.,

The research at the University of Southern California was supported by NSF under grant numbers 1409987 and 1319966.

3-consistency—is the property that each consistent assignment of values to any 2 variables has a consistent extension to any other variable. A CN is said to be strongly K-consistent iff it is k-consistent for all $k \leq K$.

The procedures that enforce local consistency have practical as well as theoretical significance. On the practical side, enforcing local consistency prunes the search space. On the theoretical side, enforcing strong K-consistency solves a CN if K is greater than or equal to the treewidth of the CN [2]. If the constraint graph of a CN with only binary constraints is a tree, then AC ensures backtrack-free search in linear time [2]. Enforcing AC is also known to solve CNs with only max-closed constraints [3]. Similarly, PC is known to ensure global consistency for CNs with only connected row convex constraints [1].

Message passing is a well-known technique for solving many combinatorial problems across a wide range of fields, such as probabilistic reasoning, artificial intelligence, statistical physics, and information theory [5, 10]. It is based on local information processing and communication, and avoids an exponential time complexity with respect to the number of variables and constraints. Although a complete theoretical analysis of its convergence and correctness is elusive, it works well in practice on many combinatorial problems such as those that arise in statistical physics, computer vision, error-correcting coding theory, or, more generally, on graphical models such as Bayesian networks and Markov random fields [10]. It has also been used to study problems such as K-satisfiability [6] and weighted constraint satisfaction [8].

Despite the significance of local consistency in constraint processing and message passing in probabilistic reasoning, the connection between them remains understudied. In this paper, we report on the close relationship between them. In light of this connection, we develop a new message passing algorithm, called the min-max message passing (MMMP) algorithm, for solving CNs. We then show how the MMMP algorithm relates to AC and how the AC-3 algorithm can be expressed more intuitively in the framework of message passing. We also show that it can be modified to enforce PC. In general, we show that the framework of message passing unifies the different concepts of local consistency.

2 Applying Message Passing to CNs

In this section, we develop a new message passing algorithm that is analogous to the standard min-sum and max-product message passing algorithms [5]. For simplicity of exposition, we assume that CNs are node-consistent. A CN can then be solved by message passing as follows. Given a CN $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, we reformulate it as computing

$$a^* = \arg\min_{a \in \mathcal{A}(\mathcal{X})} \left[E(a) \equiv \max_{C_j \in \mathcal{C}} E_{C_j}(a|S(C_j)) \right]. \tag{1}$$

Here, $\mathcal{A}(\mathcal{X}) \equiv \mathcal{D}(X_1) \times \mathcal{D}(X_2) \times \cdots \times \mathcal{D}(X_N)$ is the set of all assignments of values to all variables in $\mathcal{X}$. (For notational convenience, we also define $\mathcal{A}(\emptyset) = \{\emptyset\}$.) $a|S(C_j)$ is the projection of assignment a onto the set of variables $S(C_j)$. For α, an assignment of values to variables in $S(C_j)$, $E_{C_j}(\alpha)$ is equal to 0 if is allowed by C_j; otherwise it is equal to 1. Therefore, the CN is solvable if $\min_{a \in \mathcal{A}(\mathcal{X})} E(a) =$

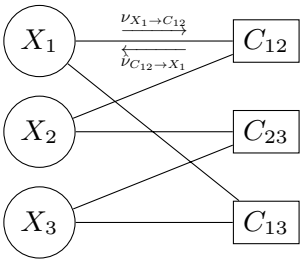

Fig. 1. Illustrates the factor graph of a CN with 3 variables $\{X_1, X_2, X_3\}$ and 3 constraints $\{C_{12}, C_{23}, C_{13}\}$. Here, $X_1, X_2 \in S(C_{12})$, $X_2, X_3 \in S(C_{23})$, and $X_1, X_3 \in S(C_{13})$. The circles represent variable vertices, and the squares represent constraint vertices. $\nu_{X_1 \to C_{12}}$ 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).

0, and its solution is any assignment a such that $E(a) = 0$; otherwise, it is not solvable. We compute a^* in Equation (1) using message passing 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 0/1 values and of size $|D(X_i)|$. Formally,

$$\nu_{X_i \to C_j} = \langle \nu_{X_i \to C_j}(X_i = x) \mid x \in \mathcal{D}(X_i) \rangle \tag{2}$$
$$\hat{\nu}_{C_j \to X_i} = \langle \hat{\nu}_{C_j \to X_i}(X_i = x) \mid x \in \mathcal{D}(X_i) \rangle. \tag{3}$$

   $\nu_{X_i \to C_j}(X_i = x)$ 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 0 and then perform update operations on them (i.e., update messages) iteratively according to

$$\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x) = \min_{a \in \mathcal{A}(\partial C_j \setminus \{X_i\})} \Big[\max \Big[E_{C_j}(a \cup \{X_i = x\}), \max_{X_k \in \partial C_j \setminus \{X_i\}} \nu^{(t-1)}_{X_k \to C_j}(a|\{X_k\}) \Big] \Big] \tag{4}$$

$$\nu^{(t)}_{X_i \to C_j}(X_i = x) = \max_{C_k \in \partial X_i \setminus \{C_j\}} \hat{\nu}^{(t)}_{C_k \to X_i}(X_i = x) \tag{5}$$

   for a chosen $X_i \in \mathcal{X}$ and $C_j \in \mathcal{C}$, and for all $x \in \mathcal{D}(X_i)$, where ∂X_i and ∂C_j are the sets of adjacent vertices of X_i and C_j in G_f, respectively; the superscript (t) indicates the update operation number; and the max operators yield 0 if they are applied on empty sets. Different from the min-sum

message passing algorithm, all summations are replaced by maximizations and no normalization constants are needed since we only care about whether the values of the variables are equal to 0. Repeat this step until convergence, i.e., Equations (4) and (5) 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 Equations (4) and (5) 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 (∞). For each variable $X_i \in \mathcal{X}$, a final assignment of value $x_i \in \mathcal{D}(X_i)$, if it exists, is given such that

$$\max_{C_j \in \partial X_i} \hat{\nu}^{(\infty)}_{C_j \to X_i}(X_i = x_i) = 0. \tag{6}$$

The set $\mathcal{D}^F_m(X_i)$ of values x_i that satisfy Equation (6) for X_i is called the *message passing domain* of X_i at the fixed point F.

Since the message update rules of Equations (4) and (5) involve only operations of minimization and maximization, we name this message passing algorithm the *min-max message passing* (MMMP) algorithm. The MMMP algorithm works analogously to other standard message passing algorithms, such as the min-sum and max-product message passing algorithms [5]. Similar to them, the MMMP algorithm neither specifies the order of the message updates in Step 3, nor provides any guarantee for the correctness of the final assignment.

We now prove that the MMMP algorithm always converges in finite time, even though convergence is not guaranteed for other message passing algorithms, such as the min-sum and max-product message passing algorithms.

Lemma 1. *No component of any message that is equal to* 1 *is changed to* 0 *by the MMMP algorithm in any update operation.*

Proof (by induction). This lemma holds trivially for the first update operation, since all components of all messages equal 0 before it.

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 $\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x) = 1$ and $\hat{\nu}^{(t+1)}_{C_j \to X_i}(X_i = x) = 0$. From Equation (4), $E_{C_j}(a \cup \{X_i = x\})$ does not change. Therefore, there must exist an $X_k \in \partial C_j \setminus \{X_i\}$ and an $x' \in \mathcal{D}(X_k)$ such that $\nu_{X_k \to C_j}(X_k = x')$ changed from 1 to 0 during the first t update operations, which contradicts the induction assumption. A similar contradiction occurs for messages from variable vertices to constraint vertices from Equation (5). The lemma continues to hold for the first $(t+1)$ update operations. □

Theorem 1. *There exists an order of message update operations such that the running time of the MMMP algorithm is bounded.*

Proof. Let the MMMP algorithm update messages in a sweeping order, i.e., messages are updated in rounds and in each round all messages are updated once. From Lemma 1, the MMMP algorithm terminates after $\mathcal{O}(d \cdot \max_{C_j \in \mathcal{C}} |S(C_j)| \cdot |\mathcal{C}|)$ rounds, where $d = \max_{X_i \in \mathcal{X}} |\mathcal{D}(X_i)|$. This upper bound measures the number of components of all messages. □

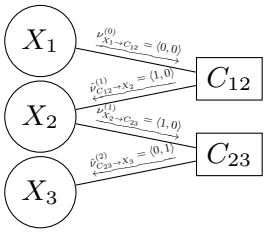

Fig. 2. Illustrates the relationship between AC and the MMMP algorithm. Here, the CN consists of 3 variables $\{X_1, X_2, X_3\}$ and 2 constraints $\{C_{12}, C_{23}\}$. Each variable has a domain of $\{0, 1\}$. C_{12} only allows $\{X_1 = 0, X_2 = 1\}$ and $\{X_1 = 1, X_2 = 1\}$; C_{23} only allows $\{X_2 = 0, X_3 = 1\}$ and $\{X_2 = 1, X_3 = 0\}$. The circles represent variable vertices, and the squares represent constraint vertices. The values of some messages are shown in the figure—the two numbers inside angle brackets are the values of the components 0 and 1, respectively. Both components of the message $\nu^{(0)}_{X_1 \to C_{12}}$ are equal to 0. Hence the message components $\hat{\nu}^{(1)}_{C_{12} \to X_2}(X_2 = 0) = 1$ and $\hat{\nu}^{(1)}_{C_{12} \to X_2}(X_2 = 1) = 0$ indicate the reduced domain $\{1\}$ that makes X_2 arc-consistent with respect to C_{12}. $\nu^{(1)}_{X_2 \to C_{23}}(X_2 = 0) = 1$ and $\nu^{(1)}_{X_2 \to C_{23}}(X_2 = 1) = 0$ indicate that X_2 should only take the value 1 while enforcing the AC of X_3 with respect to C_{23}. $\hat{\nu}^{(2)}_{C_{23} \to X_3}(X_3 = 0) = 0$ and $\hat{\nu}^{(2)}_{C_{23} \to X_3}(X_3 = 1) = 1$ indicate that under such restrictions of X_2, X_3 can only take the value 0 to be arc-consistent with respect to C_{23}.

3 Arc Consistency and the MMMP Algorithm

In this section, we study the relationship between AC and the MMMP algorithm. Consider a binary CN $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, i.e., a CN with only binary constraints. Without loss of generality, we assume that no two constraints share the same set of variables. Let C_{ij} denote the binary constraint that involves variables X_i and X_j. The factor graph representation of P therefore has two variable vertices X_i and X_j connected to each constraint vertex C_{ij}. X_i is *arc-consistent* with respect to X_j (and C_{ij}) iff for each $x_i \in \mathcal{D}(X_i)$, there exists an $x_j \in \mathcal{D}(X_j)$ such that the assignment $\{X_i = x_i, X_j = x_j\}$ is allowed in C_{ij}. P is arc-consistent iff all variables are arc-consistent with respect to each other [7]. A domain $\mathcal{D}'$ such that $\mathcal{D}'(X) \subseteq \mathcal{D}(X)$ for all $X \in \mathcal{X}$ is called an *AC domain* of P iff the CN $\langle \mathcal{X}, \mathcal{D}', \mathcal{C} \rangle$ is arc-consistent and retains all solutions of P.

Intuitively, a message from a constraint vertex C_{ij} to a variable vertex X_i encodes the AC of X_i with respect to C_{ij}, and the outgoing messages from X_i encode the prevailing values in its domain. Figure 2 illustrates this intuition.

We now formally prove a relationship between AC and the MMMP algorithm.

Theorem 2. *Under the MMMP algorithm, the CN $P' = \langle \mathcal{X}, \mathcal{D}^F_m, \mathcal{C} \rangle$ is arc-consistent for any fixed point F of any binary CN $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$.*

Proof (by contradiction). Assume that there exists a fixed point F' such that P' is not arc-consistent, i.e., there exists a constraint C_{ij} such that

$$\exists x_i \in \mathcal{D}^{F'}_m(X_i) : \forall x_j \in \mathcal{D}^{F'}_m(X_j) : \{X_i = x_i, X_j = x_j\} \notin C_{ij}. \tag{7}$$

Now consider such an x_i.

– By the definition of $E_{C_{ij}}$, from Equation (7), we have

$$\forall x_j \in \mathcal{D}_m^{F'}(X_j) : E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) = 1. \tag{8}$$

– By the definition of $\mathcal{D}_m^{F'}(X_j)$, we have

$$\forall x_j \in \mathcal{D}(X_j) \setminus \mathcal{D}_m^{F'}(X_j) : \max_{C_{jk} \in \partial X_j} \hat{\nu}^{(\infty)}_{C_{jk} \to X_j}(X_j = x_j) = 1. \tag{9}$$

Here we consider two cases for all $x_j \in \mathcal{D}(X_j) \setminus \mathcal{D}_m^{F'}(X_j)$.

- x_j satisfies $\exists C_{jk} \in \partial X_j \setminus \{C_{ij}\} : \hat{\nu}^{(\infty)}_{C_{jk} \to X_j}(X_j = x_j) = 1$. From Equation (5), we have
$$\nu^{(\infty)}_{X_j \to C_{ij}}(X_j = x_j) = 1. \tag{10}$$

- x_j satisfies $\forall C_{jk} \in \partial X_j \setminus \{C_{ij}\} : \hat{\nu}^{(\infty)}_{C_{jk} \to X_j}(X_j = x_j) = 0$. From Equation (9), this implies $\hat{\nu}^{(\infty)}_{C_{ij} \to X_j}(X_j = x_j) = 1$. By applying Equation (4) to $\hat{\nu}^{(\infty)}_{C_{ij} \to X_j}(X_j = x_j)$, we have
$$E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) = 1 \vee \nu^{(\infty)}_{X_i \to C_{ij}}(X_i = x_i) = 1. \tag{11}$$
By applying Equation (5) on $\nu^{(\infty)}_{X_i \to C_{ij}}(X_i = x_i)$, we have $\nu^{(\infty)}_{X_i \to C_{ij}}(X_i = x_i) = 0$ for the following reasons: (a) if $\partial X_i = \{C_{ij}\}$, then the max operator is applied on an empty set and thus $\nu^{(\infty)}_{X_i \to C_{ij}}(X_i = x_i) = 0$; or (b) otherwise, $\nu^{(\infty)}_{X_i \to C_{ij}}(X_i = x_i) = 1$ implies $x_i \notin D_m^{F'}(X_i)$, which contradicts the assumption. Therefore, in this case, we have
$$E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) = 1. \tag{12}$$

By applying Equation (4) to $\hat{\nu}^{(\infty)}_{C_{ij} \to X_i}(X_i = x_i)$ and plugging in Equations (8), (10) and (12), we have

$$\hat{\nu}^{(\infty)}_{C_{ij} \to X_i}(X_i = x_i) = 1, \tag{13}$$

which violates Equation (6) for $X_i = x_i$ and contradicts $x_i \in D_m^{F'}(X_i)$. □

Theorem 3. *Whenever the MMMP algorithm converges to a fixed point F on a CN P, $\mathcal{D}_m^F$ preserves all solutions of P, i.e., for any solution a, we have $\forall (X_i = x_i) \in a : x_i \in \mathcal{D}_m^F(X_i)$.*

Proof (by induction). From Lemma 1, for a variable vertex X_i, if there exists a component of a message $\hat{\nu}_{C_{ij} \to X_i}(X_i = x_i)$ that changes from 0 to 1 in some update operation, we know that $\mathcal{D}_m^F(X_i)$ does not include x_i; otherwise, $\mathcal{D}_m^F(X_i)$ includes x_i since all messages are initialized to 0. Therefore, we only need to prove that, whenever such a change occurs in the MMMP algorithm, the exclusion of x_i from $\mathcal{D}_m^F(X_i)$ preserves all solutions. We define the message passing domain of X_i after the t^{th} update operation, denoted by $\mathcal{D}_m^{(t)}(X_i)$, as the set of values x_i such that $\max_{C_{ij} \in \partial X_i} \hat{\nu}^{(t)}_{C_{ij} \to X_i}(X_i = x_i) = 0$. Upon convergence, $\mathcal{D}_m^{(t)}(X_i) = \mathcal{D}_m^F(X_i)$.

$\mathcal{D}_m^{(0)}$ preserves all solutions since all messages are initialized to 0. Assume that $\mathcal{D}_m^{(t)}$ preserves all solutions. Consider the $(t+1)^{\text{th}}$ update operation. We only consider the case where $\exists X_i \in \mathcal{X} : \mathcal{D}_m^{(t)}(X_i) \neq \mathcal{D}_m^{(t+1)}(X_i)$; otherwise, there is no solution excluded in this update operation. In this case, there exists a component of a message $\hat{\nu}_{C_{ij} \to X_i}(X_i = x_i)$ that changes from 0 to 1 in this update operation and thus $x_i \notin \mathcal{D}_m^F(X_i)$.

If $\forall x_j \in \mathcal{D}(X_j) : \nu_{X_j \to C_{ij}}^{(t)}(X_j = x_j) = 0$, then $X_i = x_i$ cannot be in any solution, since by applying Equation (4) to $\hat{\nu}_{C_{ij} \to X_i}^{(t+1)}(X_i = x_i)$ (which equals 1), we have $\forall x_j \in \mathcal{D}(X_j) : E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) = 1$. Therefore, $\exists x_j \in \mathcal{D}(X_j) : \nu_{X_j \to C_{ij}}^{(t)}(X_j = x_j) = 1$.

If $\exists x_j \in \mathcal{D}_m^{(t)}(X_j) : \nu_{X_j \to C_{ij}}^{(t)}(X_j = x_j) = 1$, from Equation (5), we have $\partial X_j \setminus \{C_{ij}\} \neq \emptyset$ and $\max_{C_{jk} \in \partial X_j \setminus \{C_{ij}\}} \hat{\nu}_{C_{jk} \to X_j}^{(t)}(X_j = x_j) = 1$, and thus by definition $x_j \notin \mathcal{D}_m^{(t)}(X_j)$, which contradicts the assumption. As a result, we have

$$\forall x_j \in \mathcal{D}_m^{(t)}(X_j) : \nu_{X_j \to C_{ij}}^{(t)}(X_j = x_j) = 0. \tag{14}$$

By applying Equation (4) to $\hat{\nu}_{C_{ij} \to X_i}^{(t+1)}(X_i = x_i)$, which equals 1, we have

$$\forall x_j \in \mathcal{D}_m^{(t)}(X_j) : \max[E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}), \nu_{X_j \to C_{ij}}^{(t)}(X_j = x_j)] = 1. \tag{15}$$

By plugging Equation (14) into the equation above, we have $\forall x_j \in \mathcal{D}_m^{(t)}(X_j) : E_{C_{ij}}(\{X_i = x_i, X_j = x_j\}) = 1$. Since $\mathcal{D}_m^{(t)}$ preserves all solutions, $X_i = x_i$ cannot be in any solution of P. □

AC can be enforced on a given CN P by AC algorithms that reduce the domains of variables without losing solutions. Arc Consistency Algorithm #3 (AC-3) is one such algorithm [4]. The graphical representation G of a CN P has vertices that represent variables and undirected edges that connect two variables iff there is a constraint involving them. AC-3 works as follows:

1. Convert G into a directed graph by replacing all edges with two arcs (directed edges) in opposite directions.
2. Insert all arcs into a queue Q.
3. Remove an arc (X_i, X_j) from Q. Ensure that X_i is arc-consistent with respect to X_j by reducing $\mathcal{D}(X_i)$, if necessary. If $\mathcal{D}(X_i)$ is reduced, insert all arcs incoming to X_i other than (X_j, X_i) into Q. Repeat this step until Q is empty.

AC-3 can be reinterpreted as an MMMP algorithm as follows.

1. Construct the factor graph for CN P.
2. Insert all messages from constraint vertices to variable vertices into a queue Q.
3. Remove a message $\hat{\nu}_{C_{ij} \to X_i}$ from Q. Update it and all messages from X_i to constraint vertices other than C_{ij} according to Equations (4) and (5), respectively. If $\nu_{X_i \to C_{ik}}$ changes, insert all messages from C_{ik} to variables other than X_i into Q. Repeat this step until Q is empty. Figure 3 illustrates one such step.

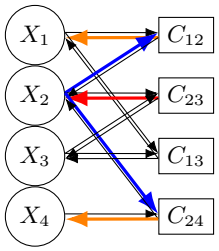

Fig. 3. Illustrates one iteration of Step 3 of MMMP-AC-3. Here, the circles represent variable vertices, and the squares represent constraint vertices. Each arrow represents a message. In this iteration, the message $\hat{\nu}_{C_{23}\to X_2}$ (red) is removed from Q and updated according to Equation (4). $\nu_{X_2\to C_{12}}$ and $\nu_{X_2\to C_{24}}$ (blue) are updated afterward according to Equation (5). Assuming that both messages change, the messages $\hat{\nu}_{C_{12}\to X_1}$ and $\hat{\nu}_{C_{24}\to X_4}$ (orange) are inserted into Q. Following the sequence of red, blue, and orange arrows, this iteration intuitively looks like paths expanded from C_{23}.

4. Reduce the domain of each variable X_i to its message passing domain.

We call this algorithm *MMMP-AC-3*. Clearly, it is an MMMP algorithm with a particular order of message update operations. Intuitively, the update of a message from a constraint vertex C_{ij} to a variable vertex X_i indicates those domain values of X_i that must be excluded to maintain the AC of X_i with respect to X_j. The update of a message from a variable vertex X_i to a constraint vertex C_{ij} indicates the reduced domain of X_i for the constraint C_{ij}.

Theorem 4. *MMMP-AC-3 terminates in bounded time.*

Proof. In Step 3, each message from a constraint vertex C_{ij} to a variable vertex X_i can be inserted into Q for at most d times, since $\nu_{X_j\to C_{ij}}$ has at most d components and thus (from Lemma 1) can change at most d times. Therefore MMMP-AC-3 has a bounded running time. □

Lemma 2. *When MMMP-AC-3 terminates, the set of values of all messages is a fixed point.*

Proof. We first prove that no message from a variable vertex to a constraint vertex changes if it is updated after MMMP-AC-3 terminates. This holds since Equation (5) holds initially and, in Step 3, each message $\nu_{X_i\to C_{ij}}$ is updated immediately after any of the messages from constraint vertices to X_i are updated.

We now prove by contradiction that no message from a constraint vertex to a variable vertex changes if it is updated after MMMP-AC-3 terminates. Assume that there exists a message $\hat{\nu}_{C_{ij}\to X_i}$ that changes if it is updated. Hence, at least one of the messages from variable vertices other than X_i to C_{ij} must have been updated after $\hat{\nu}_{C_{ij}\to X_i}$ was last updated. Therefore, $\hat{\nu}_{C_{ij}\to X_i}$ is in queue Q, which is a contradiction since MMMP-AC-3 would not have terminated then. □

Theorem 5. *MMMP-AC-3 is correct, i.e., it results in an AC domain of P.*

Proof. The theorem immediately follows from Theorems 2 to 4 and Lemma 2. □

4 Strong Path Consistency and Message Passing

In this section, we study the relationship between strong path consistency and message passing. Consider a binary CN $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$. Once again, we use C_{ij} to denote the binary constraint that involves variables X_i and X_j. X_i and X_j are *path-consistent* with respect to X_k iff, for all values $x_i \in \mathcal{D}(X_i)$ and $x_j \in \mathcal{D}(X_j)$ such that C_{ij} allows $\{X_i = x_i, X_j = x_j\}$, there exists a value $x_k \in \mathcal{D}(X_k)$ such that C_{ik} allows the assignment $\{X_i = x_i, X_k = x_k\}$ and C_{jk} allows the assignment $\{X_j = x_j, X_k = x_k\}$. A binary CN is path-consistent iff every two of its variables are path-consistent with respect to any third variable [7]. A binary CN is *strongly path-consistent* iff it is both arc-consistent and path-consistent.

Generalized message passing allows more than one intersection variable between adjacent vertices. It stems from the Kikuchi approximation of free energy in statistical mechanics [9]. The MMMP algorithm can be modified to a generalized message passing algorithm to enforce strong path consistency (strong PC). We first unify Equations (4) and (5) in the MMMP algorithm to

$$\mu^{(t)}_{U_i \to U_j}(S(U_i, U_j) = u_{ij}) = \min_{a \in \mathcal{A}(S(U_i) \setminus S(U_j))} \Big[\max \Big[E_{U_i}(a \cup \{S(U_i, U_j) = u_{ij}\}), \max_{U_k \in \partial U_i \setminus \{U_j\}} \mu^{(t-1)}_{U_k \to U_i}(a \cup \{S(U_i, U_j) = u_{ij}\} | S(U_k, U_i)) \Big] \Big], \tag{16}$$

where U_i stands for a vertex in the factor graph; $\mu_{U_i \to U_j}$ is the message sent from U_i to U_j; $S(U_i)$ stands for the scope of U_i, which is defined as

$$S(U_i) = \begin{cases} \{X_i\} & \text{if } U_i \text{ is a variable vertex } X_i, \\ S(C_i) & \text{if } U_i \text{ is a constraint vertex } C_i; \end{cases} \tag{17}$$

$S(U_i, U_j)$ stands for $S(U_i) \cap S(U_j)$; $S(U_i) = a$ stands for assigning the variables in $S(U_i)$ to a; and E_{U_i} is the potential of U_i, which is defined as

$$E_{U_i}(S(U_i) = a) = \begin{cases} 0 & \text{if } U_i \text{ is a variable vertex } X_i, \\ E_{C_i}(S(C_i) = a) & \text{if } U_i \text{ is a constraint vertex } C_i. \end{cases} \tag{18}$$

Under this unification, we can enforce strong PC by modifying the MMMP algorithm as follows. After creating the factor graph G_f, we add additional vertices (triplet vertices) with zero potentials that represent all possible combinations of 3 distinct variables, as shown in Figure 4. A triplet vertex is connected to a constraint vertex C_i iff it includes both variables in $S(C_i)$. In Step 3 of the MMMP algorithm, we replace Equations (4) and (5) by Equation (16). In Step 4 of the MMMP algorithm, we replace Equation (6) by the generalized equation

$$\max_{U_j \in \partial U_i} \mu^{(\infty)}_{U_j \to U_i}(S(U_i) = u_i | S(U_i, U_j)) = 0. \tag{19}$$

Similar to the MMMP algorithm, we call a set of values of all messages a fixed point iff it satisfies Equation (16) for all U_i in the modified factor graph. The set $\mathcal{D}^F_m(U_i)$ of values of u_i that satisfy Equation (19) for each vertex U_i is called the message passing domain of U_i at the fixed point F. The message passing domains of $U_i \in \mathcal{X}$ and $U_i \in \mathcal{C}$ are the assignments of values to each variable and each pair of variables, respectively, that enforce strong PC.

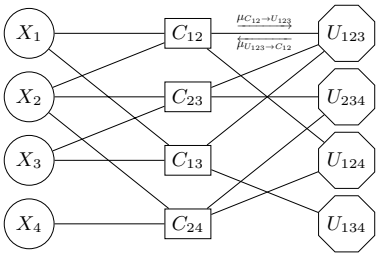

Fig. 4. Illustrates the modified factor graph for the modified MMMP algorithm. The CN consists of 4 Boolean variables $\{X_1, X_2, X_3, X_4\}$ and 4 constraints $\{C_{12}, C_{23}, C_{13}, C_{24}\}$. Here, $S(C_{12}) = \{X_1, X_2\}$, $S(C_{23}) = \{X_2, X_3\}$, $S(C_{13}) = \{X_1, X_3\}$, and $S(C_{24}) = \{X_2, X_4\}$. U_{ijk} is the triplet that represents X_i, X_j, and X_k. The circles, squares, and octagons represent variable vertices, constraint vertices, and triplet vertices, respectively. $\mu_{U_{123}\to C_{12}}$ and $\mu_{C_{12}\to U_{123}}$ are the messages from U_{123} to C_{12} and from C_{12} to U_{123}, respectively. Each of them is a vector of 0/1 values and of size $2 \times 2 = 4$. Such a pair of messages (additional to the messages shown in Figure 1) annotates each edge that is incident on a triplet vertex (even though not all are shown).

Lemma 3. *No component of any message that is equal to* 1 *is changed to* 0 *by the modified MMMP algorithm in any update operation.*

Proof. The proof is similar to that of Lemma 1. □

Theorem 6. *There exists an order of message update operations such that the running time of the modified MMMP algorithm is bounded.*

Proof. The proof is similar to that of Theorem 1. □

Theorem 7. *Under the modified MMMP algorithm, the CN* $P' = \langle \mathcal{X}, \mathcal{D}_m^F, \mathcal{C} \rangle$ *is strongly path-consistent for any fixed point* F *of any binary CN* $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$.

Proof. The proof of AC is similar to that of Theorem 2.

We now prove PC by contradiction. Let the consistent assignment of values to 2 variables $\{X_i = x_i, X_j = x_j\}$ in $\mathcal{D}_m^F(C_{ij})$ violate PC with respect to X_k (but not violate AC) in $\mathcal{D}_m^F$.

– From the PC violation assumption of $\{X_i = x_i, X_j = x_j\}$, we have

$$\forall x_k \in \mathcal{D}_m^F(X_k) : E_{C_{ik}}(\{X_i = x_i, X_k = x_k\}) = 1 \vee E_{C_{jk}}(\{X_j = x_j, X_k = x_k\}) = 1. \quad (20)$$

We use U_{ijk} to denote the triplet vertex that represents X_i, X_j, and X_k. Therefore, for such x_k's, by applying Equation (16) to $\mu^{(\infty)}_{C_{ik}\to U_{ijk}}(\{X_i = x_i, X_k = x_k\})$ and $\mu^{(\infty)}_{C_{jk}\to U_{ijk}}(\{X_j = x_j, X_k = x_k\})$, we have

$$\begin{aligned} &\forall x_k \in \mathcal{D}_m^F(X_k) : \\ &\max\left[\mu^{(\infty)}_{C_{ik}\to U_{ijk}}(\{X_i = x_i, X_k = x_k\}), \mu^{(\infty)}_{C_{jk}\to U_{ijk}}(\{X_j = x_j, X_k = x_k\})\right] = 1. \end{aligned} \quad (21)$$

– By definition of the message passing domain of X_k, we have

$$\forall x_k \in \mathcal{D}(X_k) \setminus \mathcal{D}_m^F(X_k) : \max_{C_{ik} \in \partial X_k} \mu^{(\infty)}_{C_{ik} \to X_k}(X_k = x_k) = 1. \tag{22}$$

Then, by applying Equation (16) to $\mu^{(\infty)}_{X_k \to C_{ik}}(X_k = x_k)$ and $\mu^{(\infty)}_{X_k \to C_{jk}}(X_k = x_k)$, we have

$$\forall x_k \in \mathcal{D}(X_k) \setminus \mathcal{D}_m^F(X_k) : \mu^{(\infty)}_{X_k \to C_{ik}}(X_k = x_k) = 1 \ \vee \ \mu^{(\infty)}_{X_k \to C_{jk}}(X_k = x_k) = 1. \tag{23}$$

Then by applying Equation (16) to $\mu^{(\infty)}_{C_{ik} \to U_{ijk}}(\{X_i = x_i, X_k = x_k\})$ and $\mu^{(\infty)}_{C_{jk} \to U_{ijk}}(\{X_j = x_j, X_k = x_k\})$, we have

$$\begin{aligned} &\forall x_k \in \mathcal{D}(X_k) \setminus \mathcal{D}_m^F(X_k) : \\ &\max\left[\mu^{(\infty)}_{C_{ik} \to U_{ijk}}(\{X_i = x_i, X_k = x_k\}), \mu^{(\infty)}_{C_{jk} \to U_{ijk}}(\{X_j = x_j, X_k = x_k\})\right] = 1. \end{aligned} \tag{24}$$

Applying Equation (16) to $\mu^{(\infty)}_{U_{ijk} \to C_{ij}}(\{X_i = x_i, X_j = x_j\})$, we have

$$\begin{aligned} &\mu^{(\infty)}_{U_{ijk} \to C_{ij}}(\{X_i = x_i, X_j = x_j\}) = \\ &\min_{x_k \in \mathcal{D}(X_k)} \max\left[\mu^{(\infty)}_{C_{ik} \to U_{ijk}}(\{X_i = x_i, X_k = x_k\}), \mu^{(\infty)}_{C_{jk} \to U_{ijk}}(\{X_j = x_j, X_k = x_k\})\right]. \end{aligned} \tag{25}$$

From Equations (21) and (24), we have the right-hand side of Equation (25) equal to 1. This means that $\mu_{U_{ijk} \to C_{ij}}(\{X_i = x_i, X_j = x_j\}) = 1$, which contradicts Equation (19) for C_{ij}.

Therefore, the modified MMMP algorithm enforces strong PC for P. □

Theorem 8. *Whenever the modified MMMP algorithm converges to a fixed point F on a CN P, D_m^F preserves all solutions of P, i.e., for any solution a, we have $\forall (S(U_i) = u_i) \in a : u_i \in \mathcal{D}_m^F(U_i)$.*

Proof. The proof is similar to that of Theorem 3.

Notes on the Modified MMMP Algorithm

– The running intersection property (RIP) is generally required for message passing algorithms. This property requires that, for any variable X_i, all vertices in G_f that contain it form a unique path. Ensuring the RIP for useful generalized message passing is usually complicated [9]. However, the modified MMMP algorithm does not require it, which is due to Lemma 3. Still, for any variable X_i, all vertices in the modified factor graph that contain it form a connected subgraph. We call this property the *quasi RIP*. Unlike the RIP, ensuring the quasi RIP for the modified MMMP algorithm is easy.
– The modified MMMP algorithm works even if there exist ternary constraints, in which case we simply set the potentials of the corresponding triplet vertices to represent the constraints. Thus, it can also be used to implement "generalized PC" analogous to generalized AC (GAC).

– The modified factor graph is still a bipartite graph—variable vertices and triplet vertices are in one partition, and constraint vertices are in the other partition. MMMP-AC-3 can be extended to enforce PC by treating triplet vertices as variable vertices and following the same order of message update operations[1]. This is equivalent to the Path Consistency Algorithm #2 (PC-2) [4]. The framework of the MMMP algorithm thus unifies AC-3 and PC-2.
– The modified MMMP algorithm can be extended to strong K-consistency by adding vertices that represent larger groups of variables.

5 Conclusions

We presented the MMMP algorithm for solving CNs. We established a relationship between the MMMP algorithm and AC, namely that any fixed point of the MMMP algorithm leads to an AC domain. We then showed that the AC-3 algorithm can be stated as the MMMP algorithm using a particular order of message update operations. We modified the MMMP algorithm to establish a relationship with PC. We showed that the modified MMMP algorithm has benefits that other generalized message passing algorithms do not have.

References

1. van Beek, P., Dechter, R.: On the minimality and global consistency of row-convex constraint networks. Journal of the ACM 42(3), 543–561 (1995)
2. Freuder, E.C.: A sufficient condition for backtrack-free search. Journal of the ACM 29(1), 24–32 (1982)
3. Jeavons, P.G., Cooper, M.C.: Tractable constraints on ordered domains. Artificial Intelligence 79(2), 327–339 (1995)
4. Mackworth, A.K.: Consistency in networks of relations. Artificial Intelligence 8(1), 99–118 (1977)
5. Mézard, M., Montanari, A.: Information, Physics, and Computation. Oxford University Press (2009)
6. Mézard, M., Zecchina, R.: Random k-satisfiability problem: From an analytic solution to an efficient algorithm. Physical Review E 66(5), 056126 (2002)
7. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson, 3rd edn. (2009)
8. Xu, H., Kumar, T.K.S., Koenig, S.: 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. pp. 387–402 (2017)
9. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Bethe free energy, Kikuchi approximations, and belief propagation algorithms. Tech. Rep. TR2001-16, Mitsubishi Electric Research Laboratories (2001)
10. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. Exploring Artificial Intelligence in the New Millennium 8, 239–269 (2003)

[1] We note that all messages outgoing from triplet vertices need to be initialized according to Equation (16) to ensure that Lemma 2 holds.