The Nemhauser-Trotter Reduction and Lifted Message Passing for the Weighted CSP

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. We study two important implications of the constraint composite graph (CCG) associated with the weighted constraint satisfaction problem (WCSP). First, we show that the Nemhauser-Trotter (NT) reduction popularly used for kernelization of the minimum weighted vertex cover (MWVC) problem can also be applied to the CCG of the WCSP. This leads to a polynomial-time preprocessing algorithm that fixes the optimal values of a large subset of the variables in the WCSP. Second, belief propagation (BP) is a well-known technique used for solving many combinatorial problems in probabilistic reasoning, artificial intelligence and information theory. The min-sum message passing (MSMP) algorithm is a simple variant of BP that has also been successfully employed in several research communities. Unfortunately, the MSMP algorithm has met with little success on the WCSP. We revive the MSMP algorithm for solving the WCSP by applying it on the CCG of a given WCSP instance instead of its original form. We refer to this new MSMP algorithm as the lifted MSMP algorithm for the WCSP. We demonstrate the effectiveness of our algorithms through experimental evaluations.

1 Introduction

The weighted constraint satisfaction problem (WCSP) is a combinatorial optimization problem. It is a generalization of the constraint satisfaction problem (CSP) in which the constraints are no longer "hard". Instead, each tuple in a constraint—i.e., an assignment of values to all variables in that constraint—is associated with a non-negative weight (sometimes referred to as "cost"). The goal is to find an assignment of values to all variables from their respective domains such that the total weight is minimized [1].

More formally, the WCSP is defined by a triplet $\mathcal{B} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ is a set of N variables, $\mathcal{D} = \{D_1, D_2, \ldots, D_N\}$ is a set of N domains with discrete values, and $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ is a set of M weighted constraints. Each variable $X_i \in \mathcal{X}$ can be assigned a value in its associated domain $D_i \in \mathcal{D}$. Each constraint $C_i \in \mathcal{C}$ is defined over a certain subset of the variables $S_i \subseteq \mathcal{X}$, called the scope of C_i. C_i associates a non-negative weight with each possible assignment of values to the variables in S_i.

* ORCID: 0000-0001-7874-4518

(For notational convenience, we use S_i and C_i interchangeably throughout this paper when referring to the variables participating in a weighted constraint, e.g., $X_k \in C_i \equiv X_k \in S_i$.) The goal is to find an assignment of values to all variables in $\mathcal{X}$ from their respective domains that minimizes the sum of the weights specified by each weighted constraint in $\mathcal{C}$ [1]. This combinatorial task can equivalently be characterized by having to compute

$$\arg\min_{a \in \mathcal{A}(\mathcal{X})} \sum_{C_i \in \mathcal{C}} E_{C_i}(a|C_i), \tag{1}$$

where $\mathcal{A}(\mathcal{X})$ represents the set of all $|D_1| \times |D_2| \times \ldots \times |D_N|$ complete assignments to all variables in $\mathcal{X}$. $a|C_i$ represents the projection of a complete assignment a onto the subset of variables in C_i. E_{C_i} is a function that maps each $a|C_i$ to its associated weight in C_i.

The Boolean WCSP is the WCSP in which each domain $D_i \in \mathcal{D}$ has its cardinality restricted to be 2. Despite this restriction, the Boolean WCSP is representationally as powerful as the WCSP, and it is also NP-hard to solve in general. The (Boolean) WCSP can be used to model a wide range of useful combinatorial problems arising in a large number of real-world application domains. For example, in artificial intelligence, it can be used to model user preferences [2] and combinatorial auctions. In bioinformatics, it can be used to locate RNA motifs [21]. In statistical physics, the energy minimization problem on the Potts model is equivalent to that on its corresponding pairwise Markov random field [20], which in turn can be modeled as the WCSP. In computer vision, it can be used for image restoration and panoramic image stitching [3,8].

The constraint composite graph (CCG) is a combinatorial structure associated with an optimization problem posed as the WCSP. The CCG provides a unifying framework for simultaneously exploiting the graphical structure of the variable-interactions in the WCSP as well as the numerical structure of the weighted constraints in it. The task of solving the WCSP can be reformulated as the task of finding a minimum weighted vertex cover (MWVC) on its associated CCG [9–11]. CCGs can be constructed in polynomial time and are always tripartite [9–11]. A subclass of the WCSP has instances with bipartite CCGs. This subclass is tractable since the MWVC problem can be solved in polynomial time on bipartite graphs using a staged maxflow algorithm [4].

Despite its theoretical importance, the CCG still remains largely understudied. In this paper, we study two important implications of the CCG. First, we show that the Nemhauser-Trotter (NT) reduction popularly used for kernelization of the MWVC problem [16] can also be applied to the CCG of the WCSP. This leads to a polynomial-time preprocessing algorithm that fixes the optimal values of a subset of the variables in the WCSP; and this subset is often the set of all variables. As a consequence, many WCSP instances can be solved by the polynomial-time NT reduction without search. Experimental evaluations of the NT reduction on the Boolean WCSP benchmark instances show that about $1/8^{\text{th}}$ of these benchmark instances have a kernel of size 0. In other words, about $1/8^{\text{th}}$ of these benchmark instances can be solved without search, simply by using the power of the preprocessing algorithm that encapsulates the NT reduction.

Second, belief propagation (BP) is a well-known technique used for solving many combinatorial problems in probabilistic reasoning, artificial intelligence and information theory. The min-sum message passing (MSMP) algorithm is a simple variant of BP that has also been successfully employed in several research communities. Unfortunately, the MSMP algorithm has met with little success on the WCSP. We revive the MSMP algorithm for solving the WCSP by applying it on the CCG of a given WCSP instance instead of its original form. We refer to these algorithms as the lifted MSMP algorithm (since the CCG is a lifted representation of the WCSP [9–11]) and the original MSMP algorithm, respectively. Intuitively, the lifted MSMP algorithm outperforms the original MSMP algorithm in terms of effectiveness since the CCG associated with the WCSP makes the numerical structure of its weighted constraints explicit using a tripartite graphical representation. We demonstrate the effectiveness of the lifted MSMP algorithm through experimental evaluations on the Boolean WCSP benchmark instances. We show that it outperforms the original MSMP algorithm on these benchmark instances in terms of solution quality.

2 The Constraint Composite Graph

Given an undirected graph $G = \langle V, E \rangle$, a vertex cover of G is defined as a set of vertices $S \subseteq V$ such that every edge in E has at least one of its endpoint vertices in S. A minimum vertex cover (MVC) of G is a vertex cover of minimum cardinality. When G is vertex-weighted—i.e., each vertex $v_i \in V$ has a non-negative weight w_i associated with it—its MWVC is defined as a vertex cover of minimum total weight of its vertices. The MWVC problem is to compute an MWVC on a given vertex-weighted undirected graph.

For a given graph G, the concept of the MWVC problem can be extended to the notion of projecting MWVCs onto a given independent set (IS) $U \subseteq V$. (An IS is defined as a set of vertices in which no two of them are connected by an edge.) The input to such a projection is the graph G as well as an IS $U = \{u_1, u_2, \ldots, u_k\}$. The output is a table of 2^k numbers. Each entry in this table corresponds to a k-bit vector. We say that a k-bit vector t imposes the following restrictions: (a) if the i^{th} bit t_i is 0, the vertex u_i has to be excluded from the MWVC; and (b) if the i^{th} bit t_i is 1, the vertex u_i has to be included in the MWVC. The projection of the MWVC problem onto the IS U is then defined to be a table with entries corresponding to each of the 2^k possible k-bit vectors $t^{(1)}, t^{(2)}, \ldots, t^{(2^k)}$. The value of the entry corresponding to $t^{(j)}$ is equal to the weight of the MWVC conditioned on the restrictions imposed by $t^{(j)}$. Figure 1 in [11] presents a simple example to illustrate this projection in a vertex-weighted undirected graph.

The table of numbers produced above can be viewed as a weighted constraint over $|U|$ Boolean variables. Conversely, given a (Boolean) weighted constraint, we design a lifted representation for it so as to be able to view it as the projection of MWVCs onto an IS in some intelligently constructed vertex-weighted undirected graph [9, 10]. The benefit of constructing these representations for individual

constraints lies in the fact that the lifted representation for the entire WCSP, called the CCG of the WCSP, can be obtained simply by "merging" them.

Figure 2 in [11] shows an example WCSP instance over 3 Boolean variables to illustrate the construction of the CCG. Here, there are 3 unary weighted constraints and 3 binary weighted constraints. Their lifted representations are shown next to them. The figure also illustrates how the CCG is obtained from the lifted representations of the weighted constraints: In the CCG, vertices that represent the same variable are simply "merged"—along with their edges—and every "composite" vertex is given a weight equal to the sum of the individual weights of the merged vertices. Computing the MWVC for the CCG yields a solution for the WCSP instance; namely, if X_i is in the MWVC, then it is assigned the value 1 in the WCSP instance, otherwise it is assigned the value 0 in the WCSP instance. The CCG of the WCSP can be constructed in polynomial time using the algorithms suggested in [9, 10].

3 Kernelization of the WCSP: NT Reduction on CCGs

The NT reduction is a polynomial-time kernelization procedure that reduces the size of a given MWVC problem instance [16]. It can be potentially applied on CCGs as well. It is based on the observation that the MWVC problem is a half-integral problem. This means that its Integer Linear Programming (ILP) formulation exhibits the following property. Given a graph $G = \langle V, E \rangle$, let w_i be the non-negative weight associated with vertex v_i. In the ILP formulation of the MWVC problem instance on G, a Boolean decision variable Z_i is first associated with the presence of vertex v_i in the MWVC. Then, the ILP formulation is

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{|V|} w_i Z_i, \\ \forall\ v_i \in V: \quad & Z_i \in \{0,1\}, \\ \forall\ (v_i, v_j) \in E: \quad & Z_i + Z_j \geq 1. \end{aligned} \tag{2}$$

If we relax the integrality constraints $Z_i \in \{0,1\}$ for all $i \in \{1,2,\ldots,|V|\}$ and solve the relaxed LP, the optimal solution of the LP is guaranteed to be half-integral—i.e., $\forall i \in \{1,2,\ldots,|V|\} : Z_i \in \{0, \frac{1}{2}, 1\}$. There then exists an MWVC on G that includes v_i if $Z_i = 1$ and excludes v_i if $Z_i = 0$. Therefore, one can kernelize the MWVC problem instance on G to an MWVC problem instance on a subgraph of G by retaining only those vertices whose Boolean variables in the optimal solution of the LP are $\frac{1}{2}$.

The half-integrality property can be further exploited to solve the LP relaxation of the MWVC problem with a maxflow algorithm instead of a general LP solver [4]. We first transform G to a vertex-weighted undirected bipartite graph $G_b = \langle V^L_{G_b}, V^R_{G_b}, E_{G_b} \rangle$ as follows. For each vertex $v_i \in V$, we create two vertices $v^L_i \in V^L_{G_b}$ and $v^R_i \in V^R_{G_b}$, both with weight w_i. For each edge $(v_i, v_j) \in E$, we create two edges $(v^L_i, v^R_j) \in E_{G_b}$ and $(v^L_j, v^R_i) \in E_{G_b}$. The MWVC problem can be solved in polynomial time on the bipartite graph G_b using a maxflow

algorithm [4]; and the half-integral solution of the above LP relaxation can be retrieved as follows. If both v_i^L and v_i^R are in the MWVC of G_b, then $Z_i = 1$ and v_i can be safely included in the MWVC of G; if neither v_i^L nor v_i^R is in the MWVC of G_b, then $Z_i = 0$ and v_i can be safely excluded from the MWVC of G; if exactly one of v_i^L or v_i^R is in the MWVC of G_b, then $Z_i = \frac{1}{2}$ and v_i is retained in the kernel of the MWVC problem instance posed on G.

4 Min-Sum Message Passing on the WCSP and CCGs

BP is a well-known technique for solving many combinatorial problems across a wide range of fields such as probabilistic reasoning, artificial intelligence and information theory. It can be used to solve hard inference problems 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 [20]. BP is an efficient algorithm that is based on local message passing. Although a complete theoretical analysis of its convergence and correctness is elusive, it works well in practice on many important combinatorial problems.

While BP performs message passing for the objective of marginalization over probabilities, the MSMP algorithm is a variant of BP that is used to find an assignment of values to all variables in $\boldsymbol{X}$ that minimizes functions of the form

$$E(\boldsymbol{X}) = \sum_i E_i(\boldsymbol{X}_i), \tag{3}$$

where $\boldsymbol{X}$ is the set of all variables in the global function E; E_i is a local function constituting the i^{th} term of E; and $\boldsymbol{X}_i$ is a subset of $\boldsymbol{X}$ containing all variables that participate in E_i.

To minimize the function $E(\boldsymbol{X})$, the MSMP algorithm first builds a factor graph, i.e., an undirected bipartite graph with one partition containing vertices that represent the variables in $\boldsymbol{X}$ and the other partition containing vertices that represent the local functions E_i for all i. An edge represents the participation of a variable in a local function. Furthermore, a message is associated with each direction of each edge. Intuitively, messages represent interactions between individual variables and local functions. The value of E_i is the potential of its corresponding vertex because it is indicative of its "potential" to affect other vertices. Messages are updated iteratively until convergence. In each iteration, the message from vertex u to vertex v is influenced by incoming messages to u as well as u's potential if it represents a local function. Upon convergence, a solution can be extracted from the messages.

The MSMP algorithm converges and produces an optimal solution if the factor graph is a tree [12]. This is, however, not necessarily the case if the factor graph is loopy [12]. Although the clique tree algorithm alleviates this problem to a certain extent by first converting loopy graphs to trees [7], the technique only scales to graphs with low treewidths. If the MSMP algorithm operates directly on loopy graphs, the theoretical underpinnings of its convergence and optimality properties still remain poorly understood. Nonetheless, it works well in practice

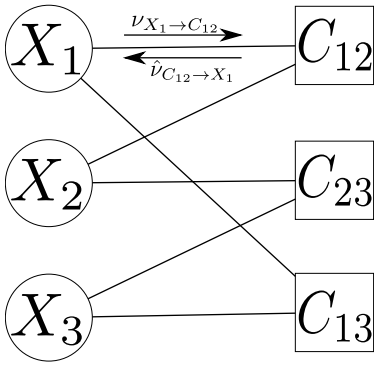


Fig. 1. Illustrates the factor graph of a Boolean WCSP instance with 3 variables $\{X_1, X_2, X_3\}$ and 3 constraints $\{C_{12}, C_{13}, C_{23}\}$. Here, $X_1, X_2 \in C_{12}$, $X_1, X_3 \in C_{13}$ and $X_2, X_3 \in C_{23}$. The circles are variable vertices, and the squares are constraint vertices. $\nu_{X_1 \to C_{12}}$ 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 (not all are explicitly shown).

on a number of important combinatorial problems in artificial intelligence, statistical physics and signal processing [12, 14]. Examples include the CSP [15], K-satisfiability [13] and the MVC problem [18]. Unfortunately, the MSMP algorithm has met with little success on the WCSP. In this section, we show how to revive the MSMP algorithm for the WCSP by using CCGs.

4.1 The MSMP Algorithm Applied Directly on the WCSP

We now describe how the MSMP algorithm can be applied directly to solve the Boolean WCSP defined by $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$. We refer to this as the original MSMP algorithm. As explained before, we first construct its factor graph. We create a vertex for each variable in $\mathcal{X}$ (variable vertex) and for each weighted constraint in $\mathcal{C}$ (constraint vertex). A variable vertex X_i and a constraint vertex C_j are connected by an edge if and only if C_j contains X_i. Figure 1 shows an example.

After the factor graph is constructed, a message (two real numbers) for each of the two directions along each edge is initialized, for instance, to zeros. A pair of messages $\nu_{X_1 \to C_{12}}$ and $\hat{\nu}_{C_{12} \to X_1}$ is illustrated in Figure 1. The messages are then updated iteratively by using the min-sum update rules given by

$$\nu^{(t)}_{X_i \to C_j}(X_i = x_i) = \sum_{C_k \in \partial X_i \setminus \{C_j\}} \left[\hat{\nu}^{(t-1)}_{C_k \to X_i}(X_i = x_i) \right] + c^{(t)}_{X_i \to C_j} \tag{4}$$

$$\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x_i) = \min_{a \in \mathcal{A}(\partial C_j \setminus \{X_i\})} \left[E_{C_j}(a | \boldsymbol{X}_j) + \sum_{X_k \in \partial C_j \setminus \{X_i\}} \nu^{(t)}_{X_k \to C_j}(a | \{X_k\}) \right] + \hat{c}^{(t)}_{C_j \to X_i} \tag{5}$$

for all $X_i \in \mathcal{X}, C_j \in \mathcal{C}$ and $x_i \in \{0, 1\}$ until convergence [12], where

- $\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x_i)$ for both $x_i \in \{0, 1\}$ are the two real numbers of the message that is passed from the constraint vertex C_j to the variable vertex X_i in the t^{th} iteration,
- $\nu^{(t)}_{X_i \to C_j}(X_i = x_i)$ for both $x_i \in \{0, 1\}$ are the two real numbers of the message that is passed from the variable vertex X_i to the constraint vertex C_j in the t^{th} iteration,
- ∂X_i and ∂C_j are the sets of neighboring vertices of X_i and C_j, respectively,
- $\boldsymbol{X}_j$ is the set of all variables in the constraint C_j, and
- $c^{(t)}_{X_i \to C_j}$ and $\hat{c}^{(t)}_{C_j \to X_i}$ are normalization constants such that

$$\min\left[\nu^{(t)}_{X_i \to C_j}(X_i = 0), \nu^{(t)}_{X_i \to C_j}(X_i = 1)\right] = 0 \tag{6}$$

$$\min\left[\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = 0), \hat{\nu}^{(t)}_{C_j \to X_i}(X_i = 1)\right] = 0. \tag{7}$$

The message update rules can be understood as follows. Each message from a variable vertex X_i to a constraint vertex C_j is updated by summing up all X_i's incoming messages from its other neighboring vertices. Each message from a constraint vertex C_j to a variable vertex X_i is updated by finding the minimum of the constraint function E_{C_j} plus the sum of all C_j's incoming messages from its other neighboring vertices. The messages can be updated in various orders.

We use the superscript ∞ to indicate the values of messages upon convergence. The final assignment of values to variables in $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ can then be found by computing

$$E_{X_i}(X_i = x_i) \equiv \sum_{C_k \in \partial X_i} \hat{\nu}^{(\infty)}_{C_k \to X_i}(X_i = x_i) \tag{8}$$

for all $X_i \in \mathcal{X}$ and $x_i \in \{0, 1\}$. Here, $E_{X_i}(X_i = 0)$ and $E_{X_i}(X_i = 1)$ can be proven to be equal to the minimum values of the total weights conditioned on $X_i = 0$ and $X_i = 1$, respectively. By selecting the value of x_i that leads to a smaller value of $E_{X_i}(X_i = x_i)$, we obtain the final assignment of values to all variables in $\mathcal{X}$.

4.2 The MSMP Algorithm Applied on CCGs

To solve a given WCSP instance, we can first transform it to an MWVC problem instance on its CCG. We can then apply the MSMP algorithm on the CCG. We refer to this procedure as the lifted MSMP algorithm.

The MWVC problem on $\langle V, E, w\rangle$—where V is the set of vertices, E is the set of edges, and w is the set of non-negative weights of the vertices—is a subclass of the Boolean WCSP. Throughout this subsection, we use the variable X_i to represent the i^{th} vertex in V: $X_i = 1$ means the i^{th} vertex is selected in the MWVC, and $X_i = 0$ means the i^{th} vertex is not selected in the MWVC. The MWVC problem can therefore be rewritten as a subclass of the Boolean WCSP with only the following two types of constraints:

– Unary weighted constraints: Each of these weighted constraints corresponds to a vertex in the MWVC problem. We use C_i^V to denote the weighted constraint that corresponds to the i^{th} vertex. C_i^V therefore only has one variable X_i. In the weighted constraint C_i^V, the tuple in which $X_i = 1$ has weight $w_i \geq 0$ and the other tuple has weight zero. This type of weighted constraints represents the minimization objective of the MWVC problem.
– Binary weighted constraints: Each of these weighted constraints corresponds to an edge in the MWVC problem. We use C_j^E to denote the weighted constraint that corresponds to the j^{th} edge. The indices of the endpoint vertices of this edge are denoted by $j(+1)$ and $j(-1)$. C_j^E therefore has two variables $X_{j(+1)}$ and $X_{j(-1)}$. In the weighted constraint C_j^E, the tuple in which $X_{j(+1)} = X_{j(-1)} = 0$ has weight infinity and the other tuples have weight zero. This type of weighted constraints represents the requirement that at least one endpoint vertex must be selected for each edge.

Given that the MWVC problem is a subclass of the Boolean WCSP, Equations 4, 5 and 8 can be reused for the MSMP algorithm on it. For the MWVC problem, these equations can be further simplified. For notational convenience, we omit normalization constants in the following derivation.

For each of the unary weighted constraints C_i^V, we have

– the added weight for selecting a vertex:

$$E_{C_i^V}(X_i = x_i) = \begin{cases} w_i & x_i = 1 \\ 0 & x_i = 0 \end{cases}, \tag{9}$$

– and exactly one variable in C_i^V:

$$\partial C_i^V \setminus \{X_i\} = \emptyset. \tag{10}$$

By plugging Equations 9 and 10 into Equation 5 for $t = \infty$, we have

$$\hat{\nu}^{(\infty)}_{C_i^V \to X_i}(X_i = x_i) = \begin{cases} w_i & x_i = 1 \\ 0 & x_i = 0 \end{cases} \tag{11}$$

for all C_i^V. Note that here we do not need Equation 4 for C_i^V since it has only one variable and thus the message passed to it does not affect the final solution.

For each of the binary weighted constraints C_j^E, we have

– the requirement that at least one endpoint vertex must be selected for each edge:

$$E_{C_j^E}(X_{j(+1)} = x_{j(+1)}, X_{j(-1)} = x_{j(-1)}) = \begin{cases} +\infty & x_{j(+1)} = x_{j(-1)} = 0 \\ 0 & \text{otherwise} \end{cases}, \tag{12}$$

– and exactly two variables in C_j^E:

$$\partial C_j^E \setminus \{X_{j(\ell)}\} = \{X_{j(-\ell)}\} \quad \forall \ell \in \{+1, -1\}. \tag{13}$$

By plugging Equations 11, 12 and 13 into Equations 4 and 5 along with the fact that there exist only unary and binary weighted constraints, we have

$$\nu^{(t)}_{X_{j(\ell)}\to C^E_j}(X_{j(\ell)}=1) = \sum_{C_k\in\partial X_{j(\ell)}\setminus\{C^V_{j(\ell)},C^E_j\}} \left[\hat{\nu}^{(t-1)}_{C_k\to X_{j(\ell)}}(X_{j(\ell)}=1)\right] + w_{j(\ell)} \tag{14}$$

$$\nu^{(t)}_{X_{j(\ell)}\to C^E_j}(X_{j(\ell)}=0) = \sum_{C_k\in\partial X_{j(\ell)}\setminus\{C^V_{j(\ell)},C^E_j\}} \hat{\nu}^{(t-1)}_{C_k\to X_{j(\ell)}}(X_{j(\ell)}=0) \tag{15}$$

$$\hat{\nu}^{(t)}_{C^E_j\to X_{j(\ell)}}(X_{j(\ell)}=1) = \min_{a\in\{0,1\}} \nu^{(t)}_{X_{j(-\ell)}\to C^E_j}(X_{j(-\ell)}=a) \tag{16}$$

$$\hat{\nu}^{(t)}_{C^E_j\to X_{j(\ell)}}(X_{j(\ell)}=0) = \nu^{(t)}_{X_{j(-\ell)}\to C^E_j}(X_{j(-\ell)}=1) \tag{17}$$

for all C^E_j and both $\ell\in\{+1,-1\}$. By plugging Equations 14 and 15 into Equations 16 and 17, we have

$$\hat{\nu}^{(t)}_{C^E_j\to X_{j(\ell)}}(X_{j(\ell)}=1) = \min_{a\in\{0,1\}}\left[\sum_{C_k\in\partial X_{j(-\ell)}\setminus\{C^V_{j(-\ell)},C^E_j\}} \left[\hat{\nu}^{(t-1)}_{C_k\to X_{j(-\ell)}}(X_{j(-\ell)}=a)\right] + w_{j(-\ell)}\cdot a\right] \tag{18}$$

$$\hat{\nu}^{(t)}_{C^E_j\to X_{j(\ell)}}(X_{j(\ell)}=0) = \sum_{C_k\in\partial X_{j(-\ell)}\setminus\{C^V_{j(-\ell)},C^E_j\}} \left[\hat{\nu}^{(t-1)}_{C_k\to X_{j(-\ell)}}(X_{j(-\ell)}=1)\right] + w_{j(-\ell)} \tag{19}$$

for all C^E_j and both $\ell\in\{+1,-1\}$, where $\hat{\nu}^{(t)}_{C^E_j\to X_{j(\ell)}}(X_{j(\ell)}=b)$ for both $b\in\{0,1\}$ are the two real numbers of the message that is passed from the j^{th} edge to the $j(\ell)^{\text{th}}$ vertex. Since each edge has exactly two endpoint vertices, the message from an edge to one of its endpoint vertices can be viewed as a message from the other endpoint vertex to it. Formally, for the j^{th} edge, we define the message from the $j(+1)^{\text{th}}$ vertex to the $j(-1)^{\text{th}}$ vertex in the t^{th} iteration as

$$\mu^{(t)}_{j(+1)\to j(-1)} \equiv \hat{\nu}^{(t)}_{C^E_j\to X_{j(-1)}}. \tag{20}$$

By plugging in Equation 20 and substituting $j(\ell)$ with i and $j(-\ell)$ with j, Equations 18 and 19 can be rewritten (with normalization constants) in the form of messages between vertices as

$$\mu^{(t)}_{j\to i}(X_i=1) = \min_{a\in\{0,1\}}\left[\sum_{k\in N(j)\setminus\{i\}} \mu^{(t-1)}_{k\to j}(X_j=a) + w_j\cdot a\right] + c^{(t)}_{j\to i} \tag{21}$$

$$\mu^{(t)}_{j\to i}(X_i=0) = \sum_{k\in N(j)\setminus\{i\}} \mu^{(t-1)}_{k\to j}(X_j=1) + w_j + c^{(t)}_{j\to i} \tag{22}$$

for all i and j such that the i^{th} and j^{th} vertices are connected by an edge in E. Here, $N(j)$ is the set of neighboring vertices of the j^{th} vertex in V and $c_{j\to i}^{(t)}$ represents the normalization constant such that $\min\left[\mu_{j\to i}^{(t)}(X_i=1), \mu_{j\to i}^{(t)}(X_i=0)\right] = 0$. Equations 21 and 22 are the message update rules of the MSMP algorithm adapted to the MWVC problem.

If the messages converge, by plugging Equations 11 and 20 into Equation 8, the final assignment of values to variables can be found by computing

$$E_{X_i}(X_i = x_i) = \sum_{j\in N(i)} \left[\mu_{j\to i}^{(\infty)}(X_i = x_i)\right] + w_i x_i, \tag{23}$$

where the meaning of $E_{X_i}(X_i = x_i)$ is similar to that in Equation 8.

5 Experimental Evaluation

In this section, we present experimental evaluations of the NT reduction and the lifted MSMP algorithm. We used two sets of Boolean WCSP benchmark instances for our experiments. The first set of benchmark instances is from the UAI 2014 Inference Competition[1]. Here, maximum a posteriori (MAP) inference queries with no evidence on the PR and MMAP benchmark instances can be reformulated as Boolean WCSP instances by first taking the negative logarithms of the probabilities in each factor and then normalizing them. The second set of benchmark instances is from [6][2]. This set includes 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). The experiments were performed on those benchmark instances that have only Boolean variables.[3]

The optimal solutions of the benchmark instances in [6] were computed using toulbar2 [6]. Since toulbar2 cannot solve WCSP instances with non-integral weights, the optimal solutions of the benchmark instances from the UAI 2014 Inference Competition were computed by finding MWVCs on their CCGs. For each benchmark instance, the MWVC problem was solved by first kernelizing it using the NT reduction, then reformulating it as an ILP [19] and finally solving the ILP using the Gurobi optimizer [5] with a running time limit of 5 minutes.

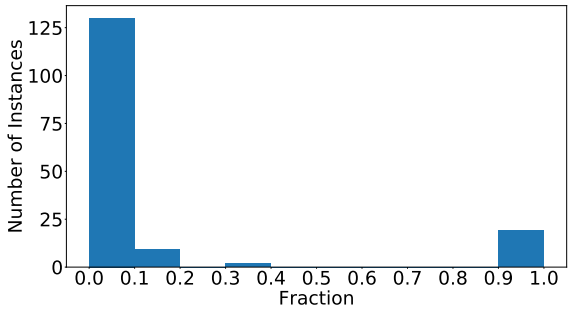
For our experiments, we implemented the NT reduction using the Gurobi optimizer [5] as the LP solver.[4] For the MSMP algorithms, we set the initial values of all messages to zeros. If no message changed by an amount more than 10^{-6}

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

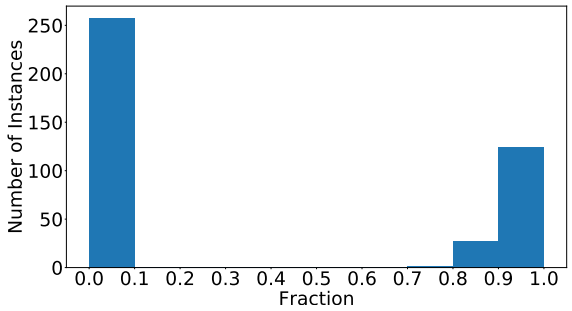
[2] http://genoweb.toulouse.inra.fr/~degivry/evalgm/

[3] As shown in [10], our techniques can also be generalized to the WCSP with larger domain sizes of the variables. However, for a proof of concept, this paper focuses on the Boolean WCSP.

[4] We could have also implemented the NT reduction using a more efficient maxflow algorithm [4]; but once again, we focus only on the proof of concept in this paper.



(a) Benchmark instances from the UAI 2014 Inference Competition: 19 out of 160 benchmark instances solved by the NT reduction

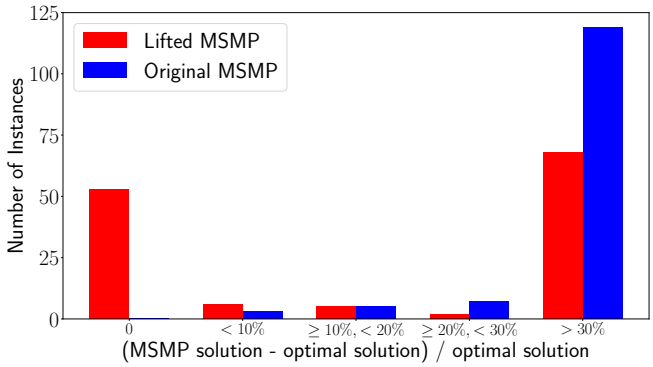


(b) Benchmark instances from [6]: 53 out of 410 benchmark instances solved by the NT reduction

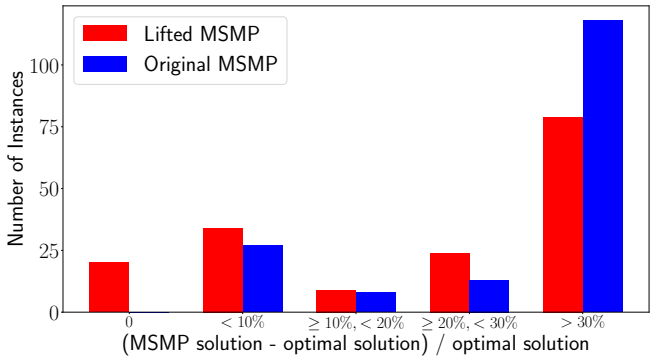
Fig. 2. Shows the effectiveness of the NT reduction. The x-axis shows the fraction of variables that are eliminated by the NT reduction. The y-axis shows the number of benchmark instances on which this happens for a fraction range.

in any iteration, we declared convergence. We used the synchronous message updating order, i.e., messages were updated in parallel in each iteration. This standardized the comparison between the two MSMP algorithms, factoring out the effects of different message updating orders within each iteration. In case of failure to converge within the time limit (5 minutes) for any benchmark instance, we reported the solution produced by the MSMP algorithm on that benchmark instance at the end of that time limit. The CCG construction algorithm and the MSMP algorithms were implemented in C++ using the Boost graph library [17] and were compiled by gcc 4.9.2 with the "-O3" option. Our experiments were performed on a GNU/Linux workstation with an Intel Xeon processor E3-1240 v3 (8MB Cache, 3.4GHz) and 16GB RAM.

Figure 2 shows the effectiveness of the NT reduction on the benchmark instances. The polynomial-time NT reduction solved about $1/8^{\text{th}}$ of these bench-



(a) Benchmark instances from the UAI 2014 Inference Competition

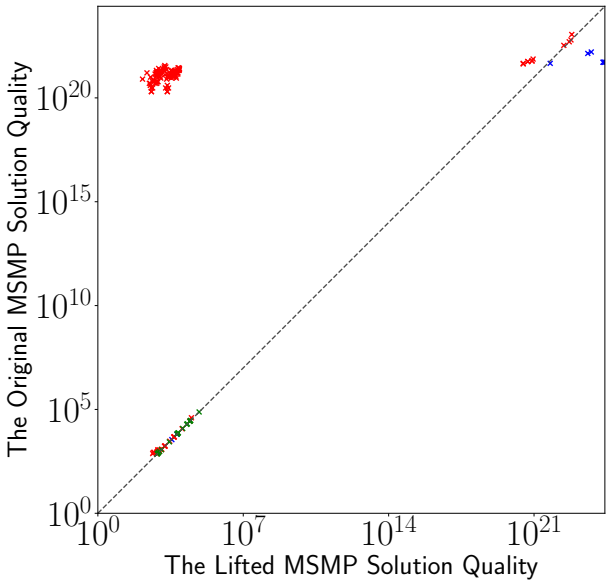


(b) Benchmark instances from [6]

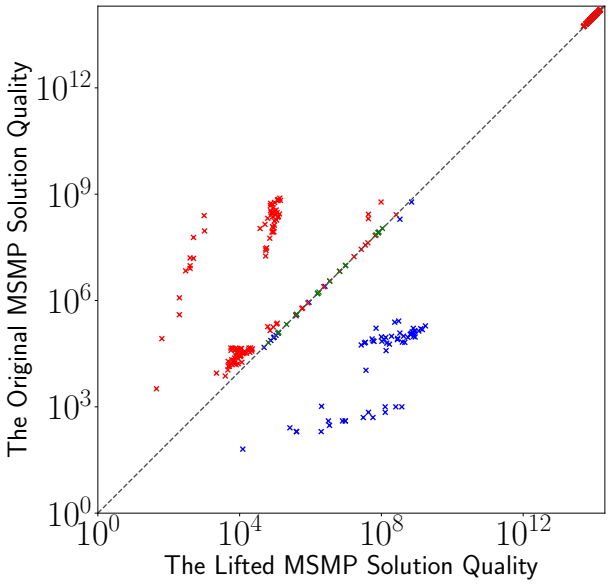
Fig. 3. Shows the qualities of the solutions (total weights) produced by the original and the lifted MSMP algorithms in comparison to the optimal solutions (for benchmark instances with known optimal solutions). The x-axis shows the suboptimality of the MSMP solutions. The y-axis shows the number of benchmark instances for a range of suboptimality. Higher bars on the left are indicative of better solutions.

mark instances yielding empty kernels. Being able to solve this many benchmark instances without search is indicative of the potential usefulness of the NT reduction for solving structured real-world problems.

Figure 3 shows the qualities of the solutions (total weights) produced by the original MSMP algorithm versus the lifted MSMP algorithm in comparison to the optimal solutions. A significant fraction of the solutions produced by the lifted MSMP algorithm are very close to the optimal solutions. However, both MSMP algorithms produced solutions that are highly suboptimal in the $> 30\%$ suboptimality range. Therefore, Figure 4 presents a direct comparison of the qualities of the solutions produced by the two MSMP algorithms. From this



(a) Benchmark instances from the UAI 2014 Inference Competition: 126/9/18 above/below/close to the diagonal dashed line



(b) Benchmark instances from [6]: 222/68/19 above/below/close to the diagonal dashed line

Fig. 4. (Caption next page.)

Fig. 4. (Figure previous page.) Shows the qualities of the solutions produced by the original MSMP algorithm in direct comparison to those produced by the lifted MSMP algorithm for both sets of benchmark instances. Each point in these plots represents a benchmark instance. The x and y coordinates of a benchmark instance represent the solution qualities produced by the lifted MSMP algorithm and the original MSMP algorithm, respectively. Benchmark instances above (red)/below (blue) the diagonal dashed line have better/worse solution qualities when using the lifted MSMP algorithm instead of the original MSMP algorithm. Benchmark instances whose MSMP solution qualities differ by only 1% are considered close (green) to the diagonal dashed line.

figure, it is evident that solution qualities of the lifted MSMP algorithm are significantly better than those of the original MSMP algorithm.

Benchmark Instance Set	Neither	Both	Original	Lifted
UAI 2014 Inference Competition	25	4	124	0
[6]	258	7	44	0

Table 1. Shows the number of benchmark instances on which each MSMP algorithm converged. The column "Neither"/"Both" indicates the number of benchmark instances on which neither/both of the MSMP algorithms converged within the time limit of 5 minutes. The column "Original"/"Lifted" indicates the number of benchmark instances on which only the original/lifted MSMP algorithm converged.

Table 1 shows the number of benchmark instances on which each MSMP algorithm converged within the time limit. Table 2 shows the convergence time and number of iterations for those benchmark instances on which both algorithms converged. Although the original MSMP algorithm converged more frequently and faster, the lifted MSMP algorithm produced better solutions in general. In addition, both MSMP algorithms are anytime and can be easily implemented in distributed settings. Therefore, the comparison of the qualities of the solutions produced is more important than that of the frequency and speed of convergence.

6 Conclusions and Future Work

We studied two important implications of the CCG associated with the WCSP. First, we showed that the NT reduction popularly used for kernelization of the MWVC problem can also be applied to the CCG of the WCSP. This leads to a polynomial-time preprocessing algorithm that fixes the optimal values of a subset of variables in a WCSP instance. This subset is often the set of all variables: We observed that the NT reduction could determine the optimal values of all variables for about $1/8^{\text{th}}$ of the benchmark instances without search.

Second, we revived the MSMP algorithm for solving the WCSP by applying it on its CCG instead of its original form. We observed not only that the lifted MSMP algorithm produced solutions that are close to optimal for a large

Benchmark Instance	The Original MSMP		The Lifted MSMP	
	Iterations	Running Time	Iterations	Running Time
U:PR/relational_2	5	0.84	9	4.00
U:PR/ra.cnf	1	0.35	6	0.34
U:PR/relational_5	5	1.18	3	0.76
U:PR/Segmentation_12	9	0.04	44	0.14
T:MRF/Segmentation/4_30_s.binary	31	0.10	60	0.13
T:MRF/Segmentation/2_28_s.binary	9	0.05	44	0.11
T:MRF/Segmentation/18_10_s.binary	15	0.07	102	0.18
T:MRF/Segmentation/12_20_s.binary	31	0.13	50	0.14
T:MRF/Segmentation/11_3_s.binary	47	0.15	176	0.24
T:MRF/Segmentation/1_28_s.binary	35	0.11	60	0.14
T:MRF/Segmentation/3_20_s.binary	31	0.12	54	0.14

Table 2. Shows the number of iterations and running time for each of the benchmark instances on which both MSMP algorithms converged within the time limit of 5 minutes. The column "Benchmark Instance" indicates the name of each benchmark instance. The "U:" and "T:" at the beginning of the names indicate that they are from the UAI 2014 Inference Competition and [6], respectively. The columns "Iterations" and "Running Time" under "The Original MSMP" and "The Lifted MSMP" indicate the number of iterations and running time (in seconds) after which the original MSMP algorithm and the lifted MSMP algorithm converged, respectively. With a few exceptions, the number of iterations and running time for the original MSMP algorithm are in general smaller than those of the lifted MSMP algorithm.

fraction of benchmark instances, but also that, in general, it produced significantly better solutions than the original MSMP algorithm. Although the lifted MSMP algorithm requires slightly more work in each iteration since the CCG is constructed using auxiliary variables, the size of the CCG is only linear in the size of the tabular representation of the WCSP [9–11], and the lifted MSMP algorithm has the benefit of producing better solutions. Both MSMP algorithms employ local message passing techniques that avoid an exponential amount of computational effort and can be readily adapted to distributed settings as well.

There are many avenues for future work. One is to extend the lifted MSMP algorithm to handle the WCSP with variables of larger domain sizes. (This extension already exists in theory [10].) Another one is to develop a distributed version of the lifted MSMP algorithm using grid/cloud computing facilities. And a third one is to explore the usefulness of constructing the CCG recursively, i.e., constructing the CCG of the MWVC problem instance on the CCG associated with a WCSP instance, and so on.

7 Acknowledgement

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1409987 and 1319966. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies or the U.S. government.

References

1. Bistarelli, S., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G., Fargier, H.: Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. Constraints 4(3), 199–240 (1999)
2. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. Journal of Artifical Intelligence Research 21, 135–191 (2004)
3. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(11), 1222–1239 (2001)
4. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. Journal of the ACM 35(4), 921–940 (1988)
5. Gurobi Optimization, Inc.: Gurobi optimizer reference manual (2016), http://www.gurobi.com
6. Hurley, B., O'Sullivan, B., Allouche, D., Katsirelos, G., Schiex, T., Zytnicki, M., de Givry, S.: Multi-language evaluation of exact solvers in graphical model discrete optimization. Constraints 21(3), 413–434 (2016)
7. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
8. Kolmogorov, V.: Primal-dual algorithm for convex Markov random fields. Tech. Rep. MSR-TR-2005-117, Microsoft Research (2005)
9. Kumar, T.K.S.: A framework for hybrid tractability results in Boolean weighted constraint satisfaction problems. In: the International Conference on Principles and Practice of Constraint Programming. pp. 282–297 (2008)
10. Kumar, T.K.S.: Lifting techniques for weighted constraint satisfaction problems. In: the International Symposium on Artificial Intelligence and Mathematics (2008)
11. Kumar, T.K.S.: Kernelization, generation of bounds, and the scope of incremental computation for weighted constraint satisfaction problems. In: the International Symposium on Artificial Intelligence and Mathematics (2016)
12. Mézard, M., Montanari, A.: Information, Physics, and Computation. Oxford University Press (2009)
13. Mézard, M., Zecchina, R.: Random k-satisfiability problem: From an analytic solution to an efficient algorithm. Physical Review E 66(5), 056126 (2002)
14. Moallemi, C.C., Roy, B.V.: Convergence of min-sum message-passing for convex optimization. IEEE Transactions on Information Theory 56(4), 2041–2050 (2010)
15. Montanari, A., Ricci-Tersenghi, F., Semerjian, G.: Solving constraint satisfaction problems through belief propagation-guided decimation. In: the Annual Allerton Conference. pp. 352–359 (2007)
16. Nemhauser, G.L., Trotter, L.E.: Vertex packings: Structural properties and algorithms. Mathematical Programming 8(1), 232–248 (1975)
17. Siek, J., Lee, L.Q., Lumsdain, A.: The Boost Graph Library: User Guide and Reference Manual. Addison-Wesley (2002)
18. Weigt, M., Zhou, H.: Message passing for vertex covers. Physical Review E 74(4), 046110 (2006)
19. Xu, H., Kumar, T.K.S., Koenig, S.: A new solver for the minimum weighted vertex cover problem. In: the International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming. pp. 392–405 (2016)

20. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. Exploring Artificial Intelligence in the New Millennium 8, 236–239 (2003)
21. Zytnicki, M., Gaspin, C., Schiex, T.: DARN! A weighted constraint solver for RNA motif localization. Constraints 13(1), 91–109 (2008)

Errata

– Equation 5 should be

$$\hat{\nu}^{(t)}_{C_j \to X_i}(X_i = x_i) = \min_{a \in \mathcal{A}(\partial C_j \setminus \{X_i\})} \left[E_{C_j}(a \cup \{X_i = x_i\}) + \sum_{X_k \in \partial C_j \setminus \{X_i\}} \nu^{(t)}_{X_k \to C_j}(a | \{X_k\}) \right] + \hat{c}^{(t)}_{C_j \to X_i}.$$

In other words, $E_{C_j}(a|\boldsymbol{X}_j)$ should be corrected to $E_{C_j}(a \cup \{X_i = x_i\})$.