A Constraint Composite Graph-Based ILP Encoding of the Boolean Weighted CSP

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

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

Abstract. The weighted constraint satisfaction problem (WCSP) occurs in the crux of many real-world applications of operations research, artificial intelligence, bioinformatics, etc. Despite its importance as a combinatorial substrate, many attempts for building an efficient WCSP solver have been largely unsatisfactory. In this paper, we introduce a new method for encoding a (Boolean) WCSP instance as an integer linear program (ILP). This encoding is based on the idea of the constraint composite graph (CCG) associated with a WCSP instance. We show that our CCG-based ILP encoding of the Boolean WCSP is significantly more efficient than previously known ILP encodings. Theoretically, we show that the CCG-based ILP encoding has a number of interesting properties. Empirically, we show that it allows us to solve many hard Boolean WCSP instances that cannot be solved by ILP solvers with previously known ILP encodings.

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 a complete assignment of values to all variables from their respective domains such that the total weight is minimized [2], called an optimal solution.

More formally, the WCSP is defined by a triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ is a set of N variables, $\mathcal{D} = \{D(X_1), D(X_2), \ldots, D(X_N)\}$ 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 $D(X_i) \in \mathcal{D}$. Each constraint $C_i \in \mathcal{C}$ is defined over a certain subset of the variables $S(C_i) \subseteq \mathcal{X}$, called the scope of C_i. C_i associates a non-negative weight with each possible assignment of values to the variables in $S(C_i)$. The goal is to find a complete assignment of values to all variables in $\mathcal{X}$ from their respective domains that minimizes the sum of the weights specified by each constraint in $\mathcal{C}$ [2]. This combinatorial task can equivalently be characterized by having to compute

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

where $A(\mathcal{X})$ represents the set of all $|D(X_1)| \times |D(X_2)| \times \ldots \times |D(X_N)|$ complete assignments to all variables in $\mathcal{X}$. $a|S(C_i)$ represents the projection of a complete assignment a onto the subset of variables in $S(C_i)$. E_{C_i} is a function that maps each $a|S(C_i)$ to its associated weight in C_i. The Boolean WCSP is the WCSP with only variables of domain size 2, i.e., $\forall X \in \mathcal{X} : |D(X)| = 2$. It is representationally as powerful as the WCSP.

There are many ways to solve a given WCSP instance. The state-of-the-art methods include best-first AND/OR search [10] and branch-and-bound algorithms that exploit soft arc consistencies [4]. Unfortunately, none of these WCSP solvers make use of the power of integer linear programming (integer LP, ILP) solvers, such as the Gurobi Optimizer [3] and lp_solve [1]. ILP solvers are highly optimized and are extensively used for solving problems in operations research. An efficient ILP encoding of the WCSP would therefore create an important nexus between constraint programming and operations research.

An ILP encoding of the WCSP can be borrowed from the probabilistic reasoning community. Here, the WCSP arises as the max-a-posteriori (MAP) problem.[1] Although this ILP encoding is popularly used in probabilistic reasoning [5, Section 13.5], it does not scale to large instances since it creates an unwieldy number of variables and constraints. In the rest of the paper, we refer to this ILP encoding as the "direct" ILP encoding.

In this paper, we introduce a new ILP encoding of the WCSP that is based on the idea of the constraint composite graph (CCG) [7–9]. We refer to this encoding as the "CCG-based" ILP encoding. We compare it with the direct ILP encoding in [5, Section 13.5] for the Boolean WCSP. We first derive and compare the theoretical bounds on the number of variables, the number of constraints and the number of variables in each constraint in the ILPs generated by these two ILP encodings. We then experimentally compare the efficiency of solving the ILPs generated by the two ILP encodings. Finally, we establish an important theoretical property of the CCG-based ILP encoding.

2 ILP Encodings of the WCSP

In this section, we describe two methods to encode a given WCSP instance as an ILP: One is the direct ILP encoding adapted from [5, Section 13.5] and the other one is our proposed CCG-based ILP encoding. For notational convenience, throughout this section, we consider the WCSP instance $\mathcal{B} = \langle \mathcal{X} = \{X_1, X_2, \ldots, X_N\}, \mathcal{D} = \{D(X_1), D(X_2), \ldots, D(X_N)\}, \mathcal{C} = \{C_1, C_2, \ldots, C_M\}\rangle$.

2.1 Direct ILP Encoding

For each $C \in \mathcal{C}$ and $a \in A(S(C))$, we introduce an ILP variable q_a^C. Here, $A(S(C))$ is the set of all assignments of values to variables in constraint C

[1] A MAP problem instance on a probabilistic graphic model, such as a Belief Network, can be formulated as a WCSP instance by taking the negative logarithm on the individual probabilities.

(therefore $|A(S(C))| = \prod_{X \in S(C)} |D(X)|$). q_a^C is either 0 or 1: If $q_a^C = 1$, then the assignment a to the variables in C is part of the to-be-determined optimal solution a^*, i.e., $a^*|S(C) = a$; otherwise it is not. The direct ILP encoding of $\mathcal{B}$ is

$$\underset{q_a^C : q_a^C \in \boldsymbol{q}}{\text{minimize}} \quad \sum_{C \in \mathcal{C}} \sum_{a \in A(S(C))} w_a^C q_a^C \tag{2}$$

$$\text{s.t.} \quad q_a^C \in \{0,1\} \quad \forall q_a^C \in \boldsymbol{q} \tag{3}$$

$$\sum_{a \in A(S(C))} q_a^C = 1 \quad \forall C \in \mathcal{C} \tag{4}$$

$$\sum_{a \in A(S(C)) : a|S(C) \cap S(C') = s} q_a^C = \sum_{a' \in A(S(C')) : a'|S(C) \cap S(C') = s} q_{a'}^{C'} \quad \forall C, C' \in \mathcal{C} \text{ and } s \in A(S(C) \cap S(C')), \tag{5}$$

where $\boldsymbol{q} = \{q_a^C \mid C \in \mathcal{C} \wedge a \in A(S(C))\}$, w_a^C denotes the weight of assignment a specified by constraint C, and $a|S(C) \cap S(C')$ is the projection of the complete assignment a onto the set of common variables in C and C'. The cardinality of $\boldsymbol{q}$ is $\sum_{C \in \mathcal{C}} \prod_{X \in S(C)} |D(X)|$. Here,

- Equation (3) represents the ILP constraints that enforce the Boolean property for all q_a^C's. It consists of $\sum_{C \in \mathcal{C}} \prod_{X \in S(C)} |D(X)| = \mathcal{O}\left(|\mathcal{C}| \hat{D}^{\hat{C}}\right)$ ILP constraints, where $\hat{C} = \max_{C \in \mathcal{C}} |S(C)|$ and $\hat{D} = \max_{X \in \mathcal{X}} |D(X)|$.
- Equation (4) represents the ILP constraints that enforce a unique assignment of values to variables in each WCSP constraint. It consists of $|\mathcal{C}|$ ILP constraints, each of which has $|A(S(C))| = \prod_{X \in S(C)} |D(X)| = \mathcal{O}\left(\hat{D}^{\hat{C}}\right)$ variables.
- Equation (5) represents the ILP constraints which enforce that every two assignments in two WCSP constraints must be consistent on their shared variables. It consists of $\mathcal{O}\left(|\mathcal{C}|^2 \hat{D}^{\hat{C}}\right)$ ILP constraints. Each of these ILP constraints has $\mathcal{O}\left(\hat{D}^{\hat{C}}\right)$ variables.

Therefore, if $\mathcal{B}$ is a Boolean WCSP instance, the direct ILP encoding has $|\boldsymbol{q}| = \mathcal{O}\left(|\mathcal{C}| \hat{D}^{\hat{C}}\right) = \mathcal{O}\left(|\mathcal{C}| 2^{\hat{C}}\right)$ variables and $\mathcal{O}\left(|\mathcal{C}|^2 \hat{D}^{\hat{C}}\right) = \mathcal{O}\left(|\mathcal{C}|^2 2^{\hat{C}}\right)$ ILP constraints. Each of these ILP constraints has $\mathcal{O}\left(\hat{D}^{\hat{C}}\right) = \mathcal{O}\left(2^{\hat{C}}\right)$ variables.

2.2 CCG-Based ILP Encoding

The CCG [7–9] is a combinatorial structure associated with the WCSP. It 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 constraints in it. The task of solving the WCSP can be reformulated as the task of finding a minimum weighted vertex cover (MWVC) (namely the MWVC problem) on its associated CCG [7–9]. CCGs can be constructed in polynomial

time and are always tripartite [7–9]. A subclass of the WCSP has instances with bipartite CCGs. This subclass is tractable since an MWVC can be found in polynomial time on bipartite graphs using a staged maxflow algorithm [6]. The CCG also enables the use of kernelization methods for the MWVC problem, such as the Nemhauser-Trotter reduction, for solving the WCSP [14]. Empirically, the min-sum message passing algorithm often produces better solutions for the MWVC problem on the CCG than directly on the WCSP [14].

We can encode a WCSP instance as an ILP after transforming it to an equivalent MWVC problem instance on its CCG $G = \langle V, E, w \rangle$. The resulting CCG-based ILP encoding is

$$\underset{x_i : v_i \in V}{\text{minimize}} \quad \sum_{i=1}^{|V|} w_i x_i \tag{6}$$

$$\text{s.t.} \quad x_i \in \{0,1\} \qquad \forall\ v_i \in V \tag{7}$$

$$x_i + x_j \geq 1 \qquad \forall\ (v_i, v_j) \in E, \tag{8}$$

where variable x_i represents the presence of v_i in the MWVC, i.e., $x_i = 1$ and $x_i = 0$ indicate that v_i is and is not in the MWVC, respectively [13]. The numbers of ILP variables and constraints are determined by the CCG. We now assume that $\mathcal{B}$ is a Boolean WCSP instance. We can compute the number of vertices and edges in the CCG by following the CCG construction procedure in [7].[2] A constraint C can be represented by the multivariate polynomial

$$\sum_{T \in \mathcal{P}(S(C))} \left[c_T \prod_{X \in T} X \right], \tag{9}$$

where $\mathcal{P}(S(C))$ is the power set of $S(C)$ and the c_T's are constants. The CCG gadget corresponding to term $c_T \prod_{X \in T} X$ has $\mathcal{O}(|T|)$ vertices and edges. The CCG gadget corresponding to constraint C therefore has an upper bound of

$$\mathcal{O}\left(\sum_{T \in \mathcal{P}(S(C))} |T| \right) = \mathcal{O}\left(\sum_{|T|=0}^{|S(C)|} \binom{|S(C)|}{|T|} |T| \right) = \mathcal{O}\left(2^{|S(C)|-1} |S(C)| \right) \tag{10}$$

vertices and edges. Therefore, if $\mathcal{B}$ is a Boolean WCSP instance, the CCG has $\mathcal{O}\left(|\mathcal{C}| 2^{\hat{C}} \hat{C} \right)$ vertices and edges constituting the ILP variables (Eq. (7)) and constraints (Eq. (8)), respectively, with each of these ILP constraints having at most 2 variables.

2.3 Comparison

We compare various parameters of the two ILP encodings for the Boolean WCSP in Table 1. For any non-trivial Boolean WCSP instances, the CCG-based ILP encoding has a huge advantage over the direct ILP encoding with respect to the

[2] As shown in [8], our techniques can also be generalized to the WCSP with variables of domain sizes larger than 2. However, for a proof of concept, this paper focuses on the Boolean WCSP.

Encoding	Direct	CCG-Based
Number of Variables	$\mathcal{O}\left(\lvert\mathcal{C}\rvert 2^{\hat{C}}\right)$	$\mathcal{O}\left(\lvert\mathcal{C}\rvert 2^{\hat{C}}\hat{C}\right)$
Number of Constraints	$\mathcal{O}\left(\lvert\mathcal{C}\rvert^2 2^{\hat{C}}\right)$	$\mathcal{O}\left(\lvert\mathcal{C}\rvert 2^{\hat{C}}\hat{C}\right)$
Number of Variables per Constraint	$\mathcal{O}\left(2^{\hat{C}}\right)$	≤ 2

Table 1: Shows the numbers of variables, constraints, and variables per constraint in the two ILP encodings of Boolean WCSP instance $\mathcal{B} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$.

number of variables per constraint. This is true even if $\hat{C}$ is bounded because, in the direct ILP encoding, the number of variables in an ILP constraint corresponding to a WCSP constraint C in Eq. (4) is $2^{|S(C)|} \geq 2$. For the number of constraints, while different values of the parameters lead to different trade-offs, the most interesting real-world applications of the WCSP have a large number $|\mathcal{C}|$ of constraints and a bounded arity $\hat{C}$ of the individual constraints. Under such assumptions, the CCG-based ILP encoding is more advantageous than the direct ILP encoding with respect to the number of constraints as well. For the number of variables, the CCG-based ILP encoding loses by a factor of $\hat{C}$. However, as argued before, in many real-world applications, $\hat{C}$ is bounded, and therefore the number of variables for both ILP encodings are of the same order. In general, when $\hat{C}$ is bounded, the CCG-based ILP encoding retains the same order of the number of variables as the direct ILP encoding and significantly wins on the number of constraints and the number of variables per constraint.

3 Experimental Evaluation

In this section, we experimentally evaluate the efficiencies of solving the Boolean WCSP using the two ILP encodings. We refer to the two algorithms that use these ILP encodings as the direct algorithm and the CCG-based algorithm.

In our experiments, the benchmark instances were generated from the UAI 2014 Inference Competition[3]. Here, MAP inference queries with no evidence on the PR and MMAP benchmark instances can be formulated as Boolean WCSP instances by first taking the negative logarithms of the probabilities in each factor and then normalizing them. The experiments were performed on those 160 benchmark instances with only Boolean variables. We set a running time limit of 120 seconds for each algorithm on each benchmark instance.

We used the Gurobi Optimizer version 7.0.2 [3] as the ILP solver. All default settings of the Gurobi Optimizer were kept except that it was configured to use only one CPU thread. The ILP encoding procedures and the CCG construction algorithm were implemented in C++ and were compiled by the GNU Compiler Collection (GCC) 6.3.0 with the "-O3" option. We used the Boost

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

Termination Status	Total	CCG-Based Only	Direct Only	Neither	Both
Number of Benchmark Instances	160	23	5	14	118

Table 2: Shows the number of benchmark instances on which the direct and CCG-based algorithms terminated within a running time limit of 120 seconds.

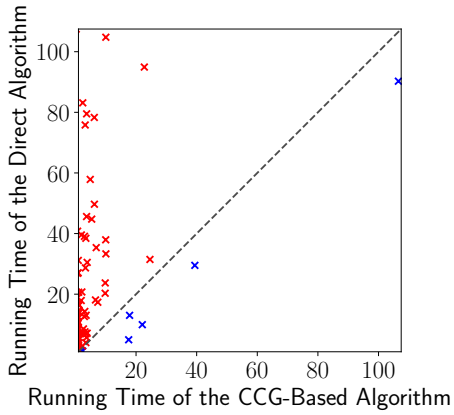

Fig. 1: Compares the efficiencies of the direct and CCG-based algorithms on the benchmark instances on which both algorithms terminated within a running time limit of 120 seconds. Each point represents a benchmark instance. The x and y coordinates of each point show the running times of the CCG-based and direct algorithm on its corresponding benchmark instance, respectively. The dashed diagonal line represents equal running times. Points above and below this line are colored red and blue, respectively. Red and blue points represent benchmark instances on which the CCG-based and direct algorithm terminated more quickly, respectively. There are 110 red points and 8 blue points.

graph library [11] to implement the graph representations and operations. We performed our experiments on a GNU/Linux workstation with an Intel Xeon processor E3-1240 v3 (8MB Cache, 3.4GHz) and 16GB RAM.

Table 2 shows the number of benchmark instances on which both algorithms terminated within the running time limit. The number of benchmark instances on which only the CCG-based algorithm terminated is much larger than the number of benchmark instances on which only the direct algorithm terminated. On 118 out of 160 benchmark instances, both terminated within the running time limit. However, even in this category, the CCG-based algorithm was much more efficient.

Figure 1 compares the efficiencies of the two algorithms on the 118 benchmark instances on which both of them terminated within the running time limit. The CCG-based algorithm was more efficient on 110 benchmark instances (red points), and the direct algorithm was more efficient on 8 benchmark instances

(blue points). Most red points are far from the dashed diagonal line, meaning that the gap between the running times of the two algorithms was very large for those benchmark instances on which the CCG-based algorithm was more efficient. On the other hand, all blue points are close to the dashed diagonal line, meaning that the direct algorithm only marginally outperformed the CCG-based algorithm on these benchmark instances in terms of running time.

4 A Theoretical Property of CCG-Based ILP Encoding

Since an ILP itself can be interpreted as a WCSP instance with an infinite weight marking the violation of an ILP constraint and unary constraints representing the ILP objective function, the concept of the CCG is well defined for ILPs. It can be constructed in polynomial time for an ILP and can be used to generate the CCG-based ILP encoding of the given ILP. A desirable property of the CCG-based ILP encoding is therefore its ability to preserve the integrality of the vertices of the feasible region of its LP relaxation.

ILPs can be relaxed to LPs by removing all integrality constraints on their variables. LPs have convex feasible regions and can therefore be solved efficiently (in polynomial time). If the feasible region of the LP relaxation of an ILP has only integer vertices (equivalent to an ILP having a totally unimodular (TUM) constraint matrix [12]), an optimal solution of the LP also yields an optimal solution of the ILP.

An ILP can be viewed as a WCSP instance as follows. Each ILP constraint translates to a WCSP constraint with weights of values zero or infinity. The ILP objective function translates to a set of unary WCSP constraints. The CCG-based ILP encoding of an ILP produces a new ILP. If the original ILP has only integer vertices in the feasible region of its LP relaxation, it is desirable for the new ILP to also have the same property. This would mean that, if the original ILP is solvable through LP relaxation, the new ILP is also solvable through LP relaxation. In this section, we show that this property of the CCG-based ILP encoding in fact holds for an important subclass of such ILPs, namely, ILPs that model MWVC problem instances on bipartite graphs.

The MWVC problem on a given vertex-weighted graph $G = \langle V, E, w\rangle$ is formulated as an ILP of the same form of Eqs. (6) to (8), where we simply associate a 0/1 variable x_i with each vertex $v_i \in V$ of non-negative weight w_i indicating the presence of v_i in the MWVC. If G is bipartite, its constraint matrix is TUM. Therefore, the LP relaxation of this ILP has only integer vertices in its feasible region [12]. We can formulate this ILP as a WCSP instance with the two types of constraints shown in Table 3.

Now we show that the CCG created for the MWVC problem on any given bipartite graph is also bipartite, which establishes that the LP relaxation of the CCG-based ILP encoding has only integer vertices in its feasible region. Consider an edge $(v_i, v_j) \in E$. The CCG gadget that represents the constraint of covering this edge involves auxiliary vertices A and A' [7]. The CCG gadget itself has the edges (v_i, A), (A, A') and (A', v_j). If the original graph is bipartite, then

x_i \ x_j	0	1
0	$+\infty$	0
1	0	0

(a) The binary constraint that represents the requirement of covering each edge $(v_i, v_j) \in E$

x_i	0	1
Value	0	w_i

(b) The unary constraint for each vertex v_i that represents a term in the objective function of minimizing the total weight of the vertex cover

Table 3: Shows the two types of WCSP constraints for the MWVC problem.

its vertices can be colored using either of two colors, red and blue, such that every edge connects a red vertex and a blue vertex. Without loss of generality, we assume that v_i is colored red and v_j is colored blue. We then color A blue and A' red. Such a coloring of the vertices ensures that the edges of the gadgets also always connect a red vertex and a blue vertex. This means that the CCG is also bipartite. Hence, we establish the desired property of the CCG-based ILP encoding for the MWVC problem on any given bipartite graph.

5 Conclusions and Future Work

In this paper, we introduced the CCG-based ILP encoding of the WCSP. We compared it to the direct ILP encoding adapted from the probabilistic reasoning community. Theoretically, we showed that the CCG-based ILP encoding has several advantages over the direct ILP encoding with respect to the number of variables per constraint and the number of constraints. Empirically, we showed that the CCG-based algorithm significantly outperforms the direct algorithm with respect to the running time on benchmark instances. Finally, we showed that MWVC problem instances on bipartite graphs, whose corresponding ILPs have only integer vertices in the feasible regions of their LP relaxations, preserve this property in their CCG-based ILP encodings as well.

It is future research to prove properties of the CCG-based ILP encoding for ILPs with TUM constraint matrices, to use our techniques to make ILP-based approaches competitive with other approaches for solving the WCSP, and to extend our results to the WCSP with variables of domain sizes larger than 2.

Acknowledgment 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. Berkelaar, M., Eikland, K., Notebaert, P.: lp_solve 5.5 open source (mixed integer) linear programming software (2004), http://lpsolve.sourceforge.net/5.5/
2. 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)
3. Gurobi Optimization, Inc.: Gurobi optimizer reference manual (2017), http://www.gurobi.com
4. 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)
5. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
6. Kumar, T.K.S.: Incremental computation of resource-envelopes in producer-consumer models. In: the International Conference on Principles and Practice of Constraint Programming. pp. 664–678 (2003)
7. 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)
8. Kumar, T.K.S.: Lifting techniques for weighted constraint satisfaction problems. In: the International Symposium on Artificial Intelligence and Mathematics (2008)
9. 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)
10. Marinescu, R., Dechter, R.: Best-first AND/OR search for graphical models. In: the AAAI Conference on Artificial Intelligence. pp. 1171–1176 (2007)
11. Siek, J., Lee, L.Q., Lumsdain, A.: The Boost Graph Library: User Guide and Reference Manual. Addison-Wesley (2002)
12. Sierksma, G.: Linear and Integer Programming: Theory and Practice. CRC Press, 2nd edn. (2001)
13. 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)
14. 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)