# Hybrid Quantum-Classical Algorithms for Solving the Weighted CSP

**Hong Xu, Kexuan Sun, Sven Koenig, Itay Hen and T. K. Satish Kumar**
University of Southern California, Los Angeles, California 90007, USA
{hongx, kexuansu, skoenig}@usc.edu,
itayhen@isi.edu, tkskwork@gmail.com

## Abstract

Many kinds of algorithms have been developed for solving the *weighted constraint satisfaction problem (WCSP)*, a combinatorial optimization problem that frequently appears in AI. Unfortunately, its NP-hardness prohibits the existence of an algorithm for solving it that is universally efficient on classical computers. Therefore, a peek into quantum computers may be imperative for solving the WCSP efficiently. In this paper, we focus on a specific type of quantum computer, called the *quantum annealer*, which approximately solves *quadratic unconstrained binary optimization (QUBO)* problems. We propose the first three *hybrid quantum-classical algorithms (HQCAs)* for the WCSP: one specifically for the binary Boolean WCSP and the other two for the general WCSP. We experimentally show that the HQCA based on simple polynomial reformulation works well on the binary Boolean WCSP, but the HQCA based on the constraint composite graph works best on the general WCSP.

## Introduction

The constraint satisfaction problem (CSP) is a well-known combinatorial optimization problem that has been used to model many important tasks in artificial intelligence (AI), such as map coloring, automated planning, scheduling and model-based diagnosis (Russell and Norvig 2009). It consists of a finite set of discrete-valued variables and constraints. Each constraint is defined over a subset of variables and consists of a finite number of tuples, each of which forbids a specific assignment of values to these variables. A solution is a complete assignment of values to all variables from their respective domains that is not forbidden by any constraint.

Despite the wide adoption of the CSP, it lacks the ability to model preferences. The weighted CSP (WCSP) (Bistarelli et al. 1999) was proposed to address this problem. It is a well-known combinatorial optimization problem that generalizes the CSP so that 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 objective is to find a complete assignment of values to all variables from their respective domains that minimizes the total weight (Bistarelli et al. 1999).

The WCSP has been used to model useful combinatorial problems for many real-world applications. For example, in AI, it has been used to model user preferences (Boutilier et al. 2004). In bioinformatics, it has been used to locate RNA motifs (Zytnicki, Gaspin, and Schiex 2008). In statistical physics, it has been used to model the energy minimization problem on the Potts model, equivalent to that on its corresponding pairwise Markov random field (Yedidia, Freeman, and Weiss 2003). In computer vision, it has been used for image restoration and panoramic image stitching (Boykov, Veksler, and Zabih 2001; Kolmogorov 2005).

Formally, the WCSP is defined by a triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, X_2, \dots, X_N\}$ is a set of $N$ variables, $\mathcal{D} = \{D(X_1), D(X_2), \dots, D(X_N)\}$ is a set of $N$ domains with discrete values, and $\mathcal{C} = \{C_1, C_2, \dots, C_M\}$ 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 objective 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 $\mathcal{C}$ (Bistarelli et al. 1999). This combinatorial optimization objective can equivalently be characterized by having to compute $\arg\min_{a \in A(\mathcal{X})} \sum_{C_i \in \mathcal{C}} E_{C_i}(a|S(C_i))$, where $A(\mathcal{X})$ represents the set of all $|D(X_1)| \times |D(X_2)| \times \ldots \times |D(X_N)|$ complete assignments of values 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 WCSP is known to be NP-hard. A WCSP instance is binary iff each constraint in it involves no more than two variables. A WCSP instance is Boolean iff each variable in it is Boolean. The binary Boolean WCSP is also NP-hard. Unfortunately, the NP-hardness of the (binary Boolean) WCSP prohibits the existence of an algorithm for solving it that is universally efficient on classical computers. Therefore, a peek into quantum computers may be imperative for solving the WCSP efficiently.

Theoretical studies in physics suggest that quantum computers are inherently more efficient than classical computers, thanks to the unique features in *quantum processes*. For example, the integer factorization problem can be solved efficiently by Shor's algorithm (Shor 1994) on quantum computers but is not known to admit an efficient classical algorithm. Among all types of quantum computer hardware, the *quantum annealer* is perhaps the most widely used type nowadays due to the commercial availability of its physical realization. The quantum annealer solves combinatorial optimization problems using a quantum process called *quantum annealing*. It has been shown that quantum annealing is more advantageous than some classical algorithms on some classes of problems (Rieffel and Polak 2014).

In reality, quantum annealing processors have only been available as the so-called "D-Wave processors" (D-W 2017). They can only take the *Ising problem*, equivalent to the *quadratic unconstrained binary*[1] *optimization (QUBO)* problem, as input. Therefore, to solve a combinatorial optimization problem other than the Ising problem, such as the WCSP, a reformulation process on classical computers is required. Such an algorithm is called a *hybrid quantum-classical algorithm (HQCA)*. Currently, there exist many HQCAs for constraint optimization problems, such as for the *maximum weighted independent set (MWIS)* problem (Choi 2008), the graph partition problem (Hen and Spedalieri 2016), the graph isomorphism problem (Hen and Sarandy 2016) and the set cover problem (Lucas 2014) as well as its generalization (Cao et al. 2016). However, since the quantum annealer is of very recent vintage, HQCAs for constraint optimization problems still remain understudied in general. Therefore, developing HQCAs for the WCSP, a very general type of constraint optimization problem, not only facilitates WCSP solving but also introduces HQCAs for other constraint optimization problems.

In this paper, we propose the first three HQCAs for approximately solving the WCSP. One HQCA is specifically for the binary Boolean WCSP based on the polynomial forms of constraints. The other two HQCAs are for the general WCSP (where there may exist non-binary constraints or non-Boolean variables), namely, one based on *integer linear programming (ILP)* and the other based on the *constraint composite graph (CCG)* (Kumar 2008a; 2008b; 2016) with our newly proposed extension to the non-Boolean WCSP. We experimentally compare these approaches and show that, while the simple polynomial reformulation works well on the binary Boolean WCSP, the CCG-based HQCA works better on the general WCSP compared to the ILP-based HQCA. These HQCAs are still far behind solvers on classical computers in terms of both their runtime and solution quality. Hence, this paper serves as a feasibility study of HQCAs for the WCSP, and we hope that they can become more useful as the quantum annealer evolves and intrigue future studies in this direction.

## Quantum Annealing

Quantum annealing is a physical process that can be used to approximately solve combinatorial optimization problems. Naively, it can be understood as a meta-heuristic algorithm that makes use of various features in quantum processes, such as *superposition*, *interference* and *entanglement*. The expected optimality gap of the solution of quantum annealing for a given problem can be theoretically analyzed, albeit requiring methods that are too sophisticated and derivation that is too complicated to be within the scope of this paper. Simply speaking, the minimum gap between the energies of the ground state and the first-excited state during quantum annealing is indicative of the suboptimality of its solution. In practice, the D-Wave processor approximately solves the Ising problem, i.e., computes

$$\operatorname*{arg\,min}_{\boldsymbol{x}=\langle x_1,x_2,\ldots,x_n\rangle} \mathcal{E}(\boldsymbol{x}) = \sum_{x_i\in\boldsymbol{x}} h_i x_i + \sum_{\{x_i,x_j\}\in\mathcal{J}} J_{ij}x_ix_j, \tag{1}$$

where $h_i$ and $J_{ij}$ are input parameters, $\boldsymbol{x} \in \{-1,+1\}^n$, and $\mathcal{J}$ is a subset of the set of all pairs of variables in $\boldsymbol{x}$ determined by the D-Wave processor. Variables $\boldsymbol{x}$ are mapped to qubits in the processor, and parameters $h_i$ and $J_{ij}$ are mapped to interactions of each qubit with the external field and every other qubit, respectively.

## Polynomial-Based HQCA

One common method for reformulating the binary Boolean WCSP as the Ising problem constructs the polynomial forms of unary and binary constraints. In this method, we first represent a Boolean variable $X_i$ in the WCSP with domain $D_i = \{0,1\}$ as $(x'_i+1)/2$, where $x'_i$ is an Ising variable with domain $\{-1,+1\}$. A unary constraint $C_i$ involving only one variable $X_i$ can then be rewritten in a polynomial form as

$$E_{C_i}(\{X_i=(x'_i+1)/2\}) = \frac{k_1+k_0}{2} + \frac{k_1-k_0}{2}x'_i, \tag{2}$$

where $k_0 = E_{C_i}(\{X_i=0\})$ and $k_1 = E_{C_i}(\{X_i=1\})$. Similarly, a binary constraint $C_{ij}$ involving two variables $X_i$ and $X_j$ can be rewritten in a polynomial form as

$$\begin{aligned} &E_{C_{ij}}(\{X_i=(x'_i+1)/2, X_j=(x'_j+1)/2\}) = \\ &c_{-1,-1} + c_{+1,-1}x'_i + c_{-1,+1}x'_j + c_{+1,+1}x'_ix'_j, \end{aligned} \tag{3}$$

where $c_{-1,-1}$, $c_{+1,-1}$, $c_{-1,+1}$ and $c_{+1,+1}$ are coefficients that can be determined by simply solving the system of linear equations

$$\begin{aligned} E_{C_{ij}}(\{X_i=0,X_j=0\}) &= c_{-1,-1} - c_{+1,-1} - c_{-1,+1} + c_{+1,+1}\\ E_{C_{ij}}(\{X_i=0,X_j=1\}) &= c_{-1,-1} - c_{+1,-1} + c_{-1,+1} - c_{+1,+1}\\ E_{C_{ij}}(\{X_i=1,X_j=0\}) &= c_{-1,-1} + c_{+1,-1} - c_{-1,+1} - c_{+1,+1}\\ E_{C_{ij}}(\{X_i=1,X_j=1\}) &= c_{-1,-1} + c_{+1,-1} + c_{-1,+1} + c_{+1,+1}. \end{aligned} \tag{4}$$

Finding an assignment of values to all $x'_i$'s so as to minimize the sum of the polynomial forms of all constraints is an Ising problem that is equivalent to solving the binary Boolean WCSP. This reformulation does not increase the input size, i.e., the number of weights in the WCSP instance equals the number of coefficients in the reformulated polynomial form. Therefore, we expect that this polynomial-based HQCA is no less advantageous than the other ones discussed below. Nevertheless, the polynomial-based HQCA is only applicable to the binary Boolean WCSP.

[1]Here, "binary" means "Boolean" in our terminology.

## ILP-Based HQCA

In this section, we address the drawbacks of the polynomial-based HQCA by introducing an ILP-based HQCA that is applicable to general WCSPs with non-binary constraints and non-Boolean variables. In the first phase, the ILP-based HQCA encodes the WCSP as an ILP problem. In the second phase, it exploits special structure in this ILP problem and casts it as an Ising problem.

The first phase is based on a direct ILP encoding of the WCSP (Xu, Koenig, and Kumar 2017), borrowed from the ILP encoding of the *map-a-posteriori* (MAP) problem in the probabilistic reasoning community (Koller and Friedman 2009, Section 13.5). For notational convenience, we assume that, for each variable $X \in \mathcal{X}$, there exists a unary constraint $C$ such that $S(C) = \{X\}$. Improving upon the ILP encoding in (Xu, Koenig, and Kumar 2017, Eqs. (2) to (5)), the ILP encoding of the WCSP is

$$\underset{q_a^C : q_a^C \in \boldsymbol{q}}{\text{minimize}} \quad \sum_{C \in \mathcal{C}} \sum_{a \in A(S(C))} w_a^C q_a^C \tag{5}$$

$$\text{s.t.} \quad q_a^C \in \{0,1\} \quad \forall q_a^C \in \boldsymbol{q} \tag{6}$$

$$\sum_{a \in A(S(C))} q_a^C = 1 \quad \forall C \in \mathcal{C} \tag{7}$$

$$\sum_{a \in A(S(C)) : a|S(C') = a'} q_a^C = q_{a'}^{C'} \tag{8}$$

$$\forall C, C' \in \mathcal{C} : |S(C')| = 1 \wedge S(C') \subset S(C), \forall a' \in A(S(C')),$$

where $\boldsymbol{q} = \{q_a^C \mid C \in \mathcal{C} \wedge a \in A(S(C))\}$, and $w_a^C$ denotes the weight of assignment $a$ specified by constraint $C$. The cardinality of $\boldsymbol{q}$ is $\sum_{C \in \mathcal{C}} \prod_{X \in S(C)} |D(X)|$. Here: (a) Equation (6) 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 $\hat{D} = \max_{X \in \mathcal{X}} |D(X)|$. (b) Equation (7) 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. (c) Equation (8) represents the ILP constraints which enforce that every two assignments in two WCSP constraints must be consistent on their shared variables. It consists of $\mathcal{O}\left(|\mathcal{C}| \cdot \hat{C} \cdot \hat{D}\right)$ ILP constraints. Each of these ILP constraints has $\mathcal{O}\left(\hat{D}^{\hat{C}-1}\right)$ variables. Unlike (Xu, Koenig, and Kumar 2017), which enforces this ILP constraint by considering every pair of WCSP constraints, here, we only consider each WCSP constraint with all its relevant unary WCSP constraints. This improvement effectively reduces the number of ILP constraints from $\mathcal{O}\left(|\mathcal{C}|^2 \hat{D}^{\hat{C}}\right)$ to $\mathcal{O}\left(|\mathcal{C}| \cdot \hat{C} \cdot \hat{D}\right)$.[2]

We now adapt the Ising formulation known for a special class of ILPs (Lucas 2014) to our case as follows. The required Ising formulation is divided into two parts

$$\min_{p_a^C : p_a^C \in \boldsymbol{p}} H = \alpha H_\alpha + \beta H_\beta, \tag{9}$$

[2]We thank an anonymous reviewer for pointing out this improvement.

where $p_a^C = 2q_a^C - 1 \in \{-1, +1\}$ and $\boldsymbol{p} = \{p_a^C \mid q_a^C \in \boldsymbol{q}\}$. Here, $H_\alpha$ represents the ILP constraints, $H_\beta$ represents the ILP optimization objective, and $\alpha$ and $\beta$ are appropriately chosen positive numbers.

For each ILP constraint, we add a squared term to $H_\alpha$ to represent it. The value of $H_\alpha$ is zero if all constraints are satisfied and positive otherwise:

$$H_\alpha = \sum_{C \in \mathcal{C}} \left[\sum_{a \in A(S(C))} q_a^C - 1\right]^2 + \sum_{\substack{C, C' \in \mathcal{C} : |S(C')| = 1 \wedge S(C') \subset S(C) \\ a' \in A(S(C'))}} \left[\sum_{a \in A(S(C)) : a|S(C') = a'} q_a^C - q_{a'}^{C'}\right]^2. \tag{10}$$

Here, after expansion of the polynomials, the quadratic terms $\left(q_a^C\right)^2$ are reduced to linear terms due to their Boolean nature, i.e., $c\left(q_a^C\right)^2 = cq_a^C$.

To capture the ILP optimization objective, we use

$$H_\beta = \sum_{C \in \mathcal{C}} \sum_{a \in A(S(C))} w_a^C q_a^C. \tag{11}$$

As far as $\alpha$ and $\beta$ are chosen to satisfy

$$\frac{\alpha}{\beta} > \sum_{C \in \mathcal{C}} \left[\sum_{a \in A(S(C))} w_a^C\right], \tag{12}$$

we are guaranteed that the minimum positive value of $\alpha H_\alpha$ is greater than the maximum value of $\beta H_\beta$, and therefore any assignment leading to a non-zero $H_\alpha$, i.e., violating at least one ILP constraint, cannot be optimal.

Combining Equations (9) to (12) and making the substitution of $q_a^C = (p_a^C + 1)/2$, we have an Ising formulation of the WCSP.

## CCG-Based HQCA

In this section, we present a CCG-based HQCA that is also applicable to general WCSPs with non-binary constraints and non-Boolean variables. In the first phase, the CCG-based HQCA encodes the WCSP as a minimum weighted vertex cover (MWVC) problem. In the second phase, it exploits special structure in the MWVC problem and casts it as an Ising problem.

The CCG (Kumar 2008a; 2008b; 2016) is a combinatorial structure associated with the Boolean WCSP. It provides a unifying framework for simultaneously exploiting the graphical structure of the variable-interactions in the Boolean WCSP as well as the numerical structure of the constraints in it. The task of solving the Boolean WCSP can be reformulated as the MWVC problem on its associated CCG (Kumar 2008a; 2008b; 2016). CCGs can be constructed in polynomial time and are always tripartite (Kumar 2008a; 2008b; 2016). The subclass of the Boolean WCSP that has instances with bipartite CCGs is tractable since an MWVC can be found in polynomial time on bipartite graphs using a staged maxflow algorithm (Kumar 2003). The CCG also enables the use of kernelization methods for the MWVC problem, such as the Nemhauser-Trotter reduction (Xu, Kumar, and Koenig 2017), for solving the Boolean WCSP. Empirically too, it is often beneficial to use the CCG in various other circumstances involving the Boolean WCSP (Xu, Kumar, and Koenig 2017; Xu, Koenig, and Kumar 2017; Fioretto et al. 2018; Yip et al. 2019).

## Constructing CCGs for the Non-Boolean WCSP

In the first phase of the algorithm, we are required to construct the CCG for the WCSP. This procedure has been extensively studied for Boolean WCSPs in (Kumar 2008a). It is very efficient and is applicable to both binary and non-binary constraints. However, the CCG construction procedure has not been extensively studied for WCSPs with non-Boolean variables, although we acknowledge the non-Boolean variable encoding from (Kumar 2008b), referred to here as the *high-degree polynomial-based encoding*. We propose three new—and more efficient—non-Boolean variable encodings, i.e., the *binary number-based encoding*, the *direct symmetric encoding*, and the *clique-based encoding*. We assume that each variable $Y_i$ has domain $D_i = \{0, 1, \ldots, d_i - 1\}$, where $d_i$ is the size of its domain.

**High-Degree Polynomial-Based Encoding** The high-degree polynomial-based encoding was first proposed in (Kumar 2008b). It uses a high-degree polynomial to represent a constraint with non-Boolean variables (as illustrated in Figure 1). Each non-Boolean variable $Y_i$ is represented by $d_i$ vertices $V_{Y_i} = \{v_{Y_i,0}, v_{Y_i,1}, \ldots, v_{Y_i,d_i-1}\}$, referred to as *variable vertices*, in the CCG gadgets. The number of these vertices in the computed MWVC indicates the value of $Y_i$.

A linear term $w \cdot Y_i$, where $w$ may be either positive or negative, can be represented by $d_i - 1$ connected components, where each connected component consists of 2 connected vertices with non-negative weights of $w_1$ and $w_2$, respectively, such that $w_1 - w_2 = w$. The vertices with weight $w_1$ represent $Y_i$. Figure 2a illustrates this.

For a negative non-linear term $-w \cdot (Y_1 \cdot Y_2 \cdot \ldots \cdot Y_m)$, where $w > 0$, we construct the CCG gadget as follows. We create a bipartite graph. The first partition contains exactly all variable vertices and assigns them weight 0. The second partition contains $\prod_{i=1}^{m}(d_i - 1)$ auxiliary vertices with weight $w$, with each of these vertices representing an assignment of values to the variables in the term. Each auxiliary vertex connects to exactly one variable vertex of each variable. It connects to variable vertices that constitute its corresponding assignment. For example, for the term $-w \cdot (Y_1 \cdot Y_2 \cdot Y_3)$, an auxiliary vertex connected to $v_{Y_{1,0}}$, $v_{Y_{2,2}}$ and $v_{Y_{3,1}}$ corresponds to the assignment $\{Y_1 = 0, Y_2 = 2, Y_3 = 1\}$. This CCG gadget represents the term $w \cdot (\prod_{i=1}^{m}(d_i - 1) - \prod_{i=1}^{m} Y_i)$. Figure 2b illustrates this. Intuitively, this can be seen as follows. $\prod_{i=1}^{m} Y_i$ auxiliary vertices have all of their incident edges covered and thus are excluded from the MWVC. That is, the total weight of the vertices selected in the MWVC is $w \cdot (\prod_{i=1}^{m}(d_i - 1) - \prod_{i=1}^{m} Y_i)$.

For a positive non-linear term $w \cdot (Y_1 \cdot Y_2 \cdot \ldots \cdot Y_m)$, where $w > 0$, we construct the CCG gadget as follows. We first create the CCG gadget as if $w < 0$. Then, to accommodate the positive coefficient, we split each edge that is adjacent to any variable vertex of $Y_1$ into two parts by inserting a vertex of a large weight $L$. These newly introduced vertices form a third partition and are meant to represent the negation of variable $Y_1$. For this reason, $L$ should be chosen to be greater than the sum of the weights of the vertices that it connects to, i.e., $L > w \cdot \prod_{i=2}^{m}(d_i - 1)$. This CCG gadget represents the term $w \cdot (\prod_{i=1}^{m}(d_i - 1) - (d_1 - 1 - Y_1) \prod_{i=2}^{m} Y_i) + L \cdot (d_1 - 1 - Y_1)$, in which the highest-degree term is the non-linear term of interest and the CCG gadgets for lower-degree terms are recursively constructed (Kumar 2008b). An illustration is shown in Figure 2c, which represents $w \cdot (18 - (3 - Y_1) \cdot Y_2 \cdot Y_3) + L \cdot (3 - Y_1)$.

**Binary Number-Based Encoding** For each non-Boolean variable $Y$ with domain size $d$, the binary number-based encoding uses $\lceil \log_2 d \rceil$ Boolean variables $\mathcal{X}_Y = \{X_{Y,1}, X_{Y,2}, \ldots, X_{Y,\lceil \log_2 d \rceil}\}$ to represent it. This converts any constraint involving this variable into a *Boolean constraint*, i.e., a constraint with only Boolean variables. The binary representation of the value of $Y$ corresponds to the values of these Boolean variables. For example, if $d = 6$, then $Y = 3$ corresponds to $X_{Y,1} = 1$, $X_{Y,2} = 1$, and $X_{Y,3} = 0$. However, if $\log_2 d$ is not an integer, then some assignments of values to variables in $\mathcal{X}_Y$ are forbidden, since they may represent values larger than what $Y$ can take. Continuing the above example, $(X_{Y,1} = 1, X_{Y,2} = 1, X_{Y,3} = 1)$ is forbidden since $Y = 7$ is not allowed. To forbid such assignments, we impose a high weight corresponding to them in the Boolean constraint. The binary number-based encoding is similar to the “log encoding” used in converting the CSP to the SAT problem (Walsh 2000).

**Direct Symmetric Encoding** For each non-Boolean variable $Y$ with domain size $d$, the direct symmetric encoding uses $d$ Boolean variables $\mathcal{X}_Y = \{X_{Y,0}, X_{Y,1}, \ldots, X_{Y,d-1}\}$ to represent it. $X_{Y,i} = 1$ and $\forall j \in \{0, 1, \ldots, d-1\} \setminus \{i\}$ : $X_{Y,j} = 0$ together indicate $Y = i$. All other assignments of values to $X_{Y,0}, X_{Y,1}, \ldots, X_{Y,d-1}$ are forbidden by a constraint, referred to as the *variable constraint*, on these $d$ variables. Constraints over non-Boolean variables are also converted to constraints over these Boolean variables representing each non-Boolean variable. This encoding is similar to the “direct encoding” used in converting the CSP to the SAT problem (Walsh 2000).

**Clique-Based Encoding** The clique-based encoding exploits the unique structure of the MWVC problem. To the best of our knowledge, this encoding does not have a counterpart in SAT encodings of the CSP. For each non-Boolean variable $Y$ with domain size $d$, the clique-based encoding uses $(d - 1)$ Boolean variables $\mathcal{X}_Y = \{X_{Y,1}, X_{Y,2}, \ldots, X_{Y,d-1}\}$ to represent it. Similar to the binary number-based and direct symmetric encodings, the clique-based encoding converts any constraint $C$ involving non-Boolean variables into a Boolean constraint $C'$. $Y = 0$ corresponds to all these Boolean variables being equal to 1, and $Y = y$, where $y \in \{1, 2, \ldots, d-1\}$, corresponds to $X_{Y,y} = 0$ and all other Boolean variables being equal to 1. All other possible assignments of values to $X_{Y,1}, X_{Y,2} \ldots, X_{Y,d-1}$ are forbidden. We impose zero weight to such variable-representationally forbidden assignments in $C'$, but forbid them with additional edges in the CCG gadget. In particular, we connect every pair of vertices representing Boolean variables in $\mathcal{X}_Y$ to form a clique. All

| $Y_i \backslash Y_j$ | 0 | 1 |
|---|---|---|
| 0 | 1 | 3 |
| 1 | 3 | 6 |
| 2 | 7 | 1 |

$$E_C(\{Y_i = y_i, Y_j = y_j\}) = c_{0,0} + c_{1,0}y_i + c_{2,0}y_i^2 + c_{0,1}y_j + c_{1,1}y_iy_j + c_{2,1}y_i^2y_j$$

$$\begin{array}{lll} E_C(\{Y_i = 0, Y_j = 0\}) = 1 & E_C(\{Y_i = 1, Y_j = 0\}) = 3 & E_C(\{Y_i = 2, Y_j = 0\}) = 7 \\ E_C(\{Y_i = 0, Y_j = 1\}) = 3 & E_C(\{Y_i = 1, Y_j = 1\}) = 6 & E_C(\{Y_i = 2, Y_j = 1\}) = 1 \end{array}$$

$$c_{0,0} = 1 \quad c_{1,0} = 1 \quad c_{2,0} = 1 \quad c_{0,1} = 2 \quad c_{1,1} = 6 \quad c_{2,1} = -5$$

Figure 1: Shows the polynomial form of the constraint $C$ on the left. The top-right panel shows the polynomial form, where $c_{0,0}$, $c_{1,0}$, $c_{2,0}$, $c_{0,1}$, $c_{1,1}$ and $c_{2,1}$ are to-be-determined coefficients. The middle-right panel shows the system of linear equations that determines all coefficients. The bottom-right panel shows the coefficients after solving the system of linear equations.

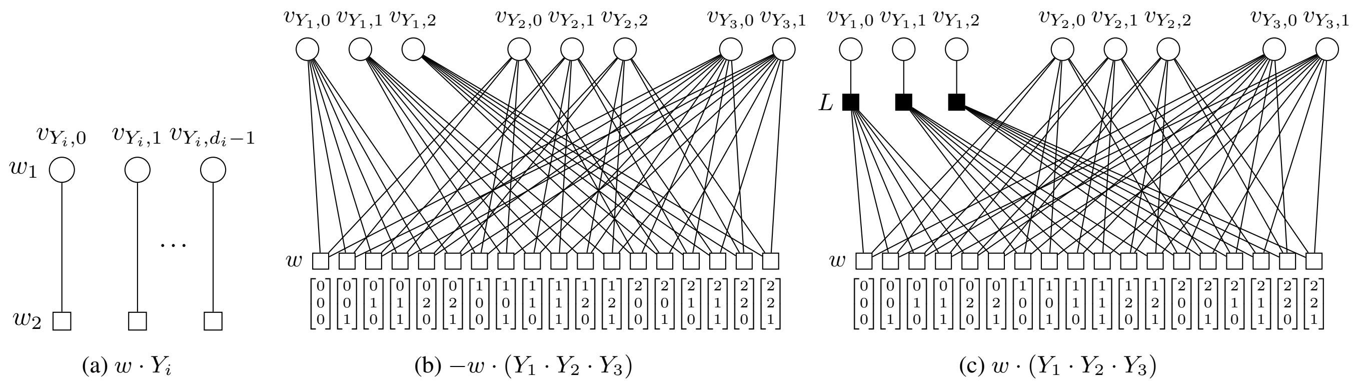

$v_{Y_i,0}$
$v_{Y_i,1}$
$v_{Y_i,d_i-1}$
$w_1$
$w_2$
(a) $w \cdot Y_i$
$v_{Y_1,0}$ $v_{Y_1,1}$ $v_{Y_1,2}$ $v_{Y_2,0}$ $v_{Y_2,1}$ $v_{Y_2,2}$ $v_{Y_3,0}$ $v_{Y_3,1}$
$w$
(b) $-w \cdot (Y_1 \cdot Y_2 \cdot Y_3)$
$L$
(c) $w \cdot (Y_1 \cdot Y_2 \cdot Y_3)$

Figure 2: Illustrates the high-degree polynomial-based encoding. In (b) and (c), the variables $Y_1$, $Y_2$ and $Y_3$ are assumed to have domain sizes 4, 4 and 3, respectively. Circles represent variable vertices. Their weights are 0 in (b) and (c) (not explicitly shown). Empty and filled squares represent the auxiliary vertices that encode the coefficients and negation of variables, respectively. The triplet of numbers below each empty square indicates the variable vertices that it connects to.

variable-representationally forbidden assignments of values to variables in $\mathcal{X}_Y$ then correspond to invalid vertex covers of the CCG gadget, while all other assignments correspond to valid vertex covers.

Consider the polynomial form of $C'$ as illustrated in Figure 1. Since all variables in $C'$ are Boolean, this polynomial is only multi-linear, i.e., the highest degree of any variable in $C'$ is 1. Furthermore, the polynomial form of $C'$ has only terms with degrees no greater than $|S(C')|$ minus the number of non-Boolean variables in $C$. This significantly simplifies the construction of the CCG gadget, since the procedure, as shown in (Kumar 2008a), to construct the CCG gadget considers each term of the polynomial one at a time. This property of the polynomial form of $C'$ is further leveraged in the clique-based encoding as follows. While the construction procedure in (Kumar 2008a) is straightforward for linear and negative non-linear terms, for a positive non-linear term $T$, we need to introduce a lower-order term $T'$ by removing a variable from $T$. To minimize the size of the resulting CCG gadget, we always choose the variable to remove from a preset order on all variables. We create this preset order by (a) fixing an order on all variables in the WCSP instance and all Boolean variables representing each of them, and (b) concatenating these groups of ordered Boolean variables according to the order of variables in $S(C)$. Compared to arbitrary choices of the variable to remove, this scheme usually results in a small number of recursively introduced lower-order terms.

**Comparing Non-Boolean Variable Encodings** We consider a constraint consisting of $n$ variables $\{Y_1, Y_2, \dots, Y_n\}$ in which in general no two weights are equal and all variables have domain size $d$. For the sake of theoretical analysis, we examine the asymptotic size of its *CCG gadget*, i.e., the CCG of a WCSP instance consisting of only this constraint, with respect to either large $n$ or large $d$ (with $n \geq 2$). The number of vertices, dominated by the number of auxiliary vertices, and the number of edges can be counted as follows (for generality, we assume that lower-order terms introduced during the construction of the CCG gadget do not cancel existing lower-order terms).

*High-Degree Polynomial-Based Encoding.* The high-degree polynomial has $n$ types of terms, where each type involves $i = 1, 2, \dots, n$ variables. The type of term that involves $i$ variables has $\binom{n}{i}$ combinations of participating variables. For $i$ given variables, there are $(d-1)^i$ terms, where each term corresponds to $\Theta\left((d-1)^i\right)$ auxiliary vertices and $\Theta\left(i(d-1)^i\right)$ edges. Therefore, the numbers of vertices and edges of the CCG gadget produced by the high-degree polynomial-based encoding are $\Theta\left(\sum_{i=1}^{n}(d-1)^{2i}\binom{n}{i}\right) = \Theta\left(\left((d-1)^2+1\right)^n\right)$ and $\Theta\left(\sum_{i=1}^{n} i(d-1)^{2i}\binom{n}{i}\right) = \Theta\left(n(d-1)^2\left((d-1)^2+1\right)^{n-1}\right)$, respectively.

*Binary Number-Based Encoding.* There are $\lceil\log_2 d\rceil$ Boolean variables representing each variable and therefore $n\lceil\log_2 d\rceil$ variables in total. By enumerating the presence and absence of these variables in each term, we have $\binom{n\lceil\log_2 d\rceil}{i}$ terms that involve $i$ variables, where each term consists of $\Theta(1)$ auxiliary vertices and $\Theta(i)$ edges. Therefore, there are $\Theta\left(\sum_{i=1}^{n\lceil\log_2 d\rceil}\binom{n\lceil\log_2 d\rceil}{i}\right) = \Theta\left(\bar{d}^n\right)$ vertices and $\Theta\left(\sum_{i=1}^{n\lceil\log_2 d\rceil} i\binom{n\lceil\log_2 d\rceil}{i}\right) = \Theta\left(n\lceil\log_2 d\rceil\bar{d}^n\right)$ edges in total, where $\bar{d} \geq d$ is the smallest integer such that $\log_2 \bar{d}$ is an integer.

*Direct Symmetric Encoding.* This encoding is similar to the so-called *one-hot encoding* popularly known in machine learning. There are $d$ Boolean variables representing each non-Boolean variable, and each Boolean constraint consists of exactly one Boolean variable representing each non-Boolean variable. Therefore, there are $d^n$ Boolean constraints of arity $n$. Now we consider the worst case, i.e., all these Boolean constraints have positive terms in their polynomial forms. For each of these Boolean constraints, we have $\mathcal{O}(n)$ auxiliary vertices and $\mathcal{O}(n^2)$ edges. Therefore, we have $\mathcal{O}(nd^n)$ vertices and $\mathcal{O}(n^2d^n)$ edges. The number of vertices and edges introduced by the variable constraint on the Boolean variables representing each non-Boolean variable can be neglected since they are only polynomial with respect to $d$. This is because the variable constraint is an *exact* $1$*-out-of-*$d$ *function*, which can be converted to $\mathcal{O}(d^2)$ binary Boolean constraints (Anthony et al. 2016). This leads to only $\mathcal{O}(nd^2)$ auxiliary vertices and edges.

*Clique-Based Encoding.* There are $(d-1)$ Boolean variables representing each non-Boolean variable, and therefore there are $n(d-1)$ Boolean variables in total. Now we consider the worst case, i.e., all terms have positive coefficients. We follow the recursive algorithm to introduce lower-order terms described in the previous subsection and, without loss of generality, assume that the non-Boolean variables are in the order $Y_n, Y_{n-1}, \dots, Y_1$. There are $\mathcal{O}\left((j+1)d^{i-1}\right)$ terms which consist of exactly $1 \leq j \leq d-1$ Boolean variables representing $Y_i$ and no Boolean variable representing $Y_{i'}$, where $i' > i$. Each term corresponds to $\Theta(1)$ auxiliary vertices and $\Theta((i-1)(d-1)+j)$ edges. Therefore, the total numbers of vertices and edges equal $\mathcal{O}\left(\sum_{i=1}^{n}\sum_{j=1}^{d-1}(j+1)d^{i-1}\right) = \mathcal{O}(d^{n+1})$ and $\mathcal{O}\left(\sum_{i=1}^{n}\sum_{j=1}^{d-1}((i-1)(d-1)+j)(j+1)d^{i-1}\right) = \mathcal{O}(nd^{n+2})$, respectively. We neglect the edges connecting Boolean variables representing each non-Boolean variable, since the number $n(d-1)(d-2)/2$ of these edges is far less than $nd^{n+2}$. Often, in practice, since it is common that some terms have negative coefficients, the number of vertices and edges can be much lower.

Table 1 summarizes the number of vertices and edges in the CCG gadget for each of the four non-Boolean variable encodings. The clique-based encoding is a preferable option. It is also preferred over the binary number-based and direct symmetric encodings because it does not introduce very large weights for encoding hard constraints and is the easiest to implement in practice. For these reasons, for the rest of this paper, we focus on the clique-based encoding for non-Boolean variables. All these non-Boolean variable encodings except for the direct symmetric encoding reduce to the same encoding for the Boolean WCSP. Despite the numbers being exponential in $n$, they are all polynomial in the actual input size, since the input size of the constraint, i.e., the number of entries in its tabular representation is $\Theta(d^n)$.

Table 1: Compares the sizes of the CCG gadgets produced by different encodings. We construct the CCG gadgets corresponding to a constraint with $n \geq 2$ variables $\{Y_1, Y_2, \dots, Y_n\}$ in which in general no two weights are equal and all variables have domain size $d$. The name of each encoding is abbreviated by its first word.

| Encoding | Vertices[†] | Edges |
|---|---|---|
| High | $\Theta\left(((d-1)^2+1)^n\right)$ | $\Theta\left(n(d-1)^2((d-1)^2+1)^{n-1}\right)$ |
| Binary | $\Theta\left(\bar{d}^n\right) = \mathcal{O}\left(2^n d^n\right)$ | $\Theta\left(n\lceil\log_2 d\rceil\bar{d}^n\right) = \mathcal{O}\left(n(\log_2 d)2^n d^n\right)$ |
| Direct | $\mathcal{O}(nd^n)$ | $\mathcal{O}(n^2d^n)$ |
| Clique | $\mathcal{O}\left(d^{n+1}\right)$ | $\mathcal{O}\left(nd^{n+2}\right)$ |

[†] By specializing the analysis to the Boolean WCSP, we note that our number of vertices is lower than that given in (Xu, Koenig, and Kumar 2017). The reason for this is that (Xu, Koenig, and Kumar 2017) counts the number of vertices in the CCG term by term in the constraint's polynomial form, which results in a loose estimate, which we avoid by counting the number of vertices constraint by constraint.

## An HQCA for the MWVC Problem

An Ising formulation of the MWVC problem on a vertex-weighted graph $G = \langle V, E, w\rangle$, adapted from the MWIS problem (Choi 2008), is as follows. For each vertex $v_i$, we associate a variable $x_i$ with it. $x_i = 0$ and $x_i = 1$ represent the presence and absence of $v_i$ in the MWVC, respectively. Then, the QUBO formulation is to minimize

$$H(x_1, x_2, \dots, x_{|V|}) = -\sum_{v_i \in V} w_i x_i + \sum_{(v_i,v_j)\in E} J_{ij} x_i x_j, \tag{13}$$

where $w_i$ is the weight associated with vertex $v_i$, and the $J_{ij}$'s satisfy $\forall (v_i, v_j) \in E : J_{ij} > \min\{w_i, w_j\}$. The minimum $H(x_1, x_2, \dots, x_{|V|})$, denoted by $H^*$, is the negative total weight of the MWIS on $G$. That is, $\sum_{v_i} w_i + H^*$ is the total weight of the MWVC on $G$ since the vertices excluded from the MWIS constitute an MWVC.

By making the substitution $x_i = (x'_i + 1)/2$ and $x_j = (x'_j + 1)/2$, where $x'_i, x'_j \in \{-1, +1\}$, we have an Ising formulation:

$$H'(x'_1, x'_2, \dots, x'_{|V|}) = -\sum_{v_i \in V} \frac{w_i}{2}(x'_i+1) + \sum_{(v_i,v_j)\in E} \frac{J_{ij}}{4}(x'_i x'_j + x'_i + x'_j + 1). \tag{14}$$

## Experimental Evaluation

We now experimentally evaluate the runtime and solution quality of these three HQCAs using a D-Wave 2X processor. It is based on a physical lattice of qubits (variables in the Ising problem) and the couplers (coefficients in the Ising problem) that connect them. These qubits and couplers together are called the *Chimera graph*, as illustrated in Figure 3. In a D-Wave 2X processor (or any other currently available D-Wave processor), the Chimera graph is sparse. Therefore, it may not be possible to feed many Ising problem instances with dense connectivity directly into the D-Wave 2X processor. In this case, the process of *embedding* is indispensable, which is to find an equivalent Ising problem instance that can be directly fed into the D-Wave 2X processor. In our experiments, for a proof of concept, we simply used

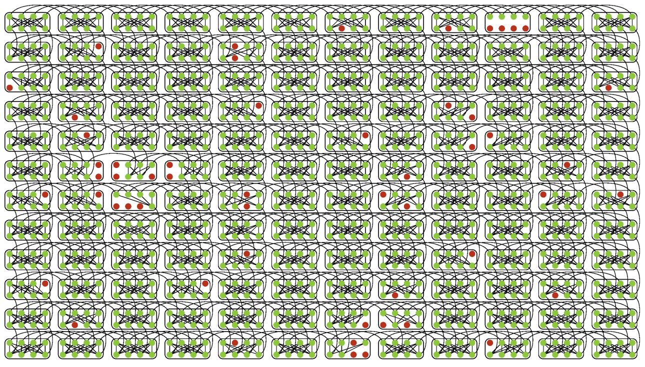
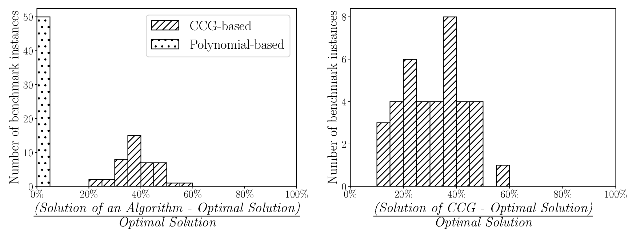


Figure 3: Shows the Chimera graph in a D-Wave 2X processor. The Chimera graph consists of a lattice of "imperfect" $K_{4,4}$ bipartite graph units. The green dots represent qubits, and edges represent couplers. The red dots represent missing qubits in the $K_{4,4}$ units.



(a) (b)

Figure 4: Compares the suboptimalities of the solutions produced by HQCAs on the two benchmark instance sets. The x-axes show ranges of suboptimalities of the solutions produced by the CCG-based and the polynomial-based HQCAs. The y-axes show the number of benchmark instances in a range of suboptimalities. The left plot compares the qualities of the solutions produced by the CCG-based and the polynomial-based HQCAs with optimal solutions on the first benchmark instance set with only Boolean variables. The polynomial-based HQCA produced optimal solutions on 39 out of 50 benchmark instances and the suboptimalities on all other benchmark instances are smaller than 10%. The right plot compares the suboptimalities of the solutions produced by the CCG-based HQCA with optimal solutions on the second benchmark instance set with non-Boolean variables. While some of the optimality gaps may look large, they are not particularly uncommon for HQCAs due to the peculiar uncertainty nature of quantum annealers (e.g., (Bian et al. 2016, Figure 3)).

the D-Wave library (D-W 2017) to find embeddings. This library always finds an equivalent problem instance (assuming that no bug is triggered).

In our experiments, we selected real-world benchmark instances from the industrial weighted partial Max-SAT category of the Eleventh Max-SAT Evaluation[3] and reformulated them as Boolean WCSP instances. We selected benchmark instances whose number of variables is less than 30 to accommodate the limited number of qubits of the D-Wave 2X processor. Only two benchmark instances (`wcsp/spot5/dir/8.wcsp.dir.wcnf` and `wcsp/spot5/log/8.wcsp.log.wcnf`) satisfy this criterion. The polynomial-based HQCA is not applicable to them. In addition, our experiments showed that the ILP-based HQCA could not embed any of them within the 5-minute time limit. The solution costs produced by the CCG-based HQCA are 96 and 5, respectively, while the optimal solution costs for both benchmark instances are 2.

[3]http://www.maxsat.udl.cat/16/benchmarks/index.html

Popular real-world benchmark instances, such as those in the Eleventh Max-SAT evaluation and those used in (Hurley et al. 2016), are too large to be embedded into the D-Wave 2X processor. For this reason, we also generated two random WCSP benchmark instance sets. (HQCAs that work on the D-Wave 2X processor will also work on larger benchmark instances on more advanced quantum annealing processors in the future.) In each benchmark instance in the first benchmark instance set, the number of variables is 50 and all variables are Boolean; for each pair of variables, we generated a binary constraint between them with probability $p = 0.13$. We assigned a random integer weight between 0 and 100 to each tuple in these constraints. In each benchmark instance in the second benchmark instance set, the number of variables is 20 and the domain size of each variable is randomly set to be 2 or 3. Constraints are generated in the same way as in the first benchmark instance set, except $p = 0.2$. Given the way the benchmark instances were generated, the average number of constraints that each variable participates in is about 3. We used the functions `find_embedding` and `unembed_answer` from the D-Wave Python library to find embeddings and restore solutions to the original benchmark instances, respectively. For `find_embedding`, we set the runtime limit to 1000 seconds and turned the `fast_embedding` option on for trading off fast embedding against embedding quality. For each benchmark instance, we requested the D-Wave 2X processor to run 1000 times (while the embedding procedure only needs to be run once)[4]. For all benchmark instances, we also obtained optimal solutions using `toulbar2` (Hurley et al. 2016), a state-of-the-art WCSP solver that always produces optimal solutions. Each benchmark instance was solved by `toulbar2` within 1 second (the minimum runtime that can be measured by `toulbar2`). The process of solving Ising instances was performed on a D-Wave 2X processor while other processes, including finding embeddings and restoring solutions by the `unembed` procedure, were performed on a GNU/Linux workstation with an Intel Xeon processor E3-1240 v3 (8MB Cache, 3.4GHz) and 16GB RAM.

Figure 4 compares the qualities of the solutions produced by the polynomial-based and CCG-based HQCAs with the optimal solutions on the benchmark instances from both benchmark instance sets. The ILP-based HQCA could not embed any benchmark instances into the Chimera graph within the time limit and is thus not shown. For the first benchmark instance set, as expected, the polynomial-based HQCA was the most advantageous. The CCG-based HQCA successfully embedded 43 out of 50 benchmark instances

[4]While this may seem unusual for algorithms on classical computers, it is common practice to run the quantum annealing procedure for thousands of times (as it normally terminates very quickly).

between 10 and 354 seconds. For the second benchmark instance set, where the polynomial-based HQCA is not applicable, the CCG-based HQCA successfully embedded 38 out of 50 benchmark instances between 3 and 292 seconds. Despite the 1000 runs, the runtime of the D-Wave 2X processor on each benchmark instance is within 450 milliseconds. The Ising formulation processes in both HQCAs ran within 60 milliseconds. The majority of the time was consumed by the functions `find_embedding` and `unembed_answer`. Although these HQCAs cannot compete with `toulbar2`, in fact, if `find_embedding` and `unembed_answer` are not required, the efficiency and effectiveness of HQCAs can be outstanding for approximately solving the Boolean WCSP. To verify this, we generated 50 Ising problem instances, which can be seen as special cases of Boolean WCSP instances, by randomly selecting 50% of the edges of the Chimera graph as constraints with random integer weights. We used the D-Wave 2X processor and `toulbar2` to solve them. We also reformulated them as weighted Max-SAT instances and solved them using `open-wbo` (Martins, Manquinho, and Lynce 2014). The experimental results showed that the quantum annealer produced solutions within 400 milliseconds and the qualities of the solutions were better than those produced by both `toulbar2` and `open-wbo` within a 5-minute runtime limit.

## Conclusions and Future Work

In this paper, we proposed the first three HQCAs for solving the WCSP, namely, the polynomial-based, ILP-based and CCG-based HQCAs. For the CCG-based HQCA, we acknowledged one and proposed three new non-Boolean variable encodings to extend the applicability of the CCG to the non-Boolean WCSP. We evaluated the HQCAs using experiments on a D-Wave 2X processor, a physical realization of the quantum annealer. We showed that the polynomial-based HQCA works well on the binary Boolean WCSP, but the CCG-based HQCA is not only more widely applicable but also works better than the ILP-based HQCA on the general WCSP (where the polynomial-based HQCA is not applicable). While these HQCAs are still far behind solvers, such as `toulbar2`, on classical computers with respect to both runtime and solution quality, we hope that these HQCAs become more useful as the quantum annealer evolves and that they can serve as a starting point for future developments in using the quantum annealer for solving the WCSP.

One direction of future work is to improve the CCG-based HQCA by utilizing kernelization methods, such as the Nemhauser-Trotter reduction (Nemhauser and Trotter 1975) and the crown reduction (Chlebík and Chlebíková 2008), so as to reduce the size of the CCG in polynomial time. Another direction is to integrate state-of-the-art WCSP solvers on classical computers and the quantum annealer. Yet another direction is to develop efficient CCG representations dedicated to more specialized types of constraint optimization problems, such as weighted Max-SAT and weighted Max-Cut problems. Finally, it is also important to understand HQCAs for the WCSP with more theoretical rigor.

## Acknowledgements

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189 and 1837779. 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

Anthony, M.; Boros, E.; Crama, Y.; and Gruber, A. 2016. Quadratization of symmetric pseudo-Boolean functions. *Discrete Applied Mathematics* 203:1–12.

Bian, Z.; Chudak, F.; Israel, R. B.; Lackey, B.; Macready, W. G.; and Roy, A. 2016. Mapping constrained optimization problems to quantum annealing with application to fault diagnosis. *Frontiers in ICT* 3(14).

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

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artifical Intelligence Research* 21:135–191.

Boykov, Y.; Veksler, O.; and Zabih, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(11):1222–1239.

Cao, Y.; Jiang, S.; Perouli, D.; and Kais, S. 2016. Solving set cover with pairs problem using quantum annealing. *Scientific Reports* 6(33957).

Chlebík, M., and Chlebíková, J. 2008. Crown reductions for the minimum weighted vertex cover problem. *Discrete Applied Mathematics* 156(3):292–312.

Choi, V. 2008. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing* 7(5):193–209.

D-Wave Systems Inc. 2017. *Developer Guide for Python (09-1024A-F).*

Fioretto, F.; Xu, H.; Koenig, S.; and Kumar, T. K. S. 2018. Solving multiagent constraint optimization problems on the constraint composite graph. In *the International Conference on Principles and Practice of Multi-Agent Systems*, 106–122.

Hen, I., and Sarandy, M. S. 2016. Driver Hamiltonians for constrained optimization in quantum annealing. *Physical Review A* 93(6):062312.

Hen, I., and Spedalieri, F. M. 2016. Quantum annealing for constrained optimization. *Physical Review Applied* 5(3):034007.

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

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

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

Kumar, T. K. S. 2003. Incremental computation of resource-envelopes in producer-consumer models. In *the International Conference on Principles and Practice of Constraint Programming*, 664–678.

Kumar, T. K. S. 2008a. A framework for hybrid tractability results in Boolean weighted constraint satisfaction problems. In *the International Conference on Principles and Practice of Constraint Programming*, 282–297.

Kumar, T. K. S. 2008b. Lifting techniques for weighted constraint satisfaction problems. In *the International Symposium on Artificial Intelligence and Mathematics*.

Kumar, T. K. S. 2016. Kernelization, generation of bounds, and the scope of incremental computation for weighted constraint satisfaction problems. In *the International Symposium on Artificial Intelligence and Mathematics*.

Lucas, A. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2:5.

Martins, R.; Manquinho, V.; and Lynce, I. 2014. Open-WBO: A modular MaxSAT solver. In *the International Conference on Theory and Applications of Satisfiability Testing*, 438–445.

Nemhauser, G. L., and Trotter, L. E. 1975. Vertex packings: Structural properties and algorithms. *Mathematical Programming* 8(1):232–248.

Rieffel, E. G., and Polak, W. H. 2014. *Quantum Computing: A Gentle Introduction*. MIT Press.

Russell, S., and Norvig, P. 2009. *Artificial Intelligence: A Modern Approach*. Pearson, 3rd edition.

Shor, P. W. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In *the Annual Symposium on Foundations of Computer Science*, 124–134.

Walsh, T. 2000. SAT v CSP. In *the International Conference on Principles and Practice of Constraint Programming*, 441–456.

Xu, H.; Koenig, S.; and Kumar, T. K. S. 2017. A constraint composite graph-based ILP encoding of the Boolean weighted CSP. In *the International Conference on Principles and Practice of Constraint Programming*, 630–638.

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

Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2003. Understanding belief propagation and its generalizations. *Exploring Artificial Intelligence in the New Millennium* 8:236–239.

Yip, K. W.; Xu, H.; Koenig, S.; and Kumar, T. K. S. 2019. Quadratization of nonlinear pseudo-Boolean functions via the constraint composite graph. In *the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research.*

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