Lifting Techniques for Weighted Constraint Satisfaction Problems

T. K. Satish Kumar
Computer Science Division
University of California, Berkeley
tksk@eecs.berkeley.edu

Abstract

In this paper, we identify rich tractable classes of *Weighted Constraint Satisfaction Problems (WCSPs)*. Our results stem from employing a set of transformation techniques—referred to as "*Lifting*"—that considers each constraint *locally*. We show that, in general, WCSPs are reducible to *minimum weighted vertex cover problems* in *tripartite graphs*; and many tractable classes of WCSPs can be recognized by their reducibility to *minimum weighted vertex cover problems* in *bipartite graphs*. We examine the implications of our approach when combined with other mathematical tools, and provide a framework for tightly characterizing the complexity of solving a given *instance* of the WCSP.

1 Introduction

In many real-life problem domains, we are required to express natural factors like fuzziness, probabilities, preferences and/or costs, and are subsequently interested in finding an optimal solution with respect to one or more criteria. Towards this end, many extensions to the basic CSP model have been introduced to incorporate non-crisp constraints, probabilities, weights, etc. These include many variants like *Fuzzy CSPs*, *Probabilistic CSPs* and *Weighted CSPs*.[1]

Roughly speaking, a WCSP is a generalization of a CSP in which the constraints are no longer "hard", but are extended by associating non-negative *costs* to the tuples. The goal is then to find an assignment of values to all the variables from their respective domains so that the *total cost* is *minimized*. More formally, a 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 *variables*, and $\mathcal{C} = \{C_1, C_2 \ldots C_M\}$ is a set of *weighted constraints* between the variables. Each variable X_i is associated with a discrete-valued *domain* $D_i \in \mathcal{D}$, and each weighted constraint C_i is defined on a certain subset $S_i \subseteq \mathcal{X}$ of the variables. S_i is referred to as the *scope* of C_i; and C_i specifies a non-negative *cost* for each possible combination of values to the variables in S_i. An *optimal* solution is an assignment of values to all the variables from their respective domains so that the *sum* of the costs (as specified locally by each of the weighted constraints) is *minimized*. It is well known that, in general, optimally solving WCSPs is NP-hard.

Representationally, WCSPs can model numerous important combinatorial problems arising in many different application domains; examples include (but are not limited to) representing and reasoning about user preferences (Boutilier *et al* 2004), planning with goal preferences (Do *et al* 2007), resource allocation, combinatorial auctions, and bioinformatics. Quite importantly, WCSPs also arise as *Energy Minimization Problems (EMPs)* in probabilistic settings. EMPs are fundamental to many important applications; in computer vision, for example, tasks such as image restoration, total variation minimization and panoramic image stitching can be formulated as EMPs derived in the context of Markov Random Fields (MRFs) (Kolmogorov 2005).[2]

In this paper, we identify several rich tractable classes of WCSPs. Our results stem from employing a set of transformation techniques—referred to as "*Lifting*"—that considers each constraint only *locally*. We show that, in general, WCSPs are reducible to *minimum weighted vertex cover problems* in *tripartite graphs*; and many tractable classes of WCSPs can be recognized by their reducibility to *minimum weighted vertex cover problems* in *bipartite graphs*. Our approach yields very simple arguments for establishing the tractability of several interesting classes of WCSPs that were: (a) previously known to be tractable, and (b) not previously known to be tractable—e.g., classes of WCSPs with general domain sizes of the variables and/or general arities of the weighted constraints. We examine the implications of our approach when combined with other mathematical tools, and provide a framework for tightly characterizing the complexity of solving a given *instance* of the WCSP.

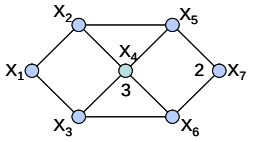
2 Background Results in Graph Theory

In this section, we will briefly review some fundamental results in graph theory, and set up the groundwork for the rest of the paper. In later sections, we will study the relevance of these results in the context of solving WCSPs.

Given an undirected graph $G = \langle V, E \rangle$, a *matching* is a subset of edges $M \subseteq E$ such that no two edges in M share a common end-point. A *maximum matching* is a matching

[1] These in turn can be viewed as particular instances of certain meta-frameworks like *Valued CSPs* (Schiex *et al* 1995) and/or *Semiring-based CSPs* (Bistarelli *et al* 1996).

[2] Here, the minimum Energy setting corresponds to a *maximum a-posteriori* labeling of the variables.



X_1 \ X_4	0	1
0	4	7
1	5	6

Figure 1: The left-hand side shows a node-weighted undirected graph. The weights on X_4 and X_7 are set to 3 and 2 respectively; and all other weights are assumed to be 1. The projection of the minimum weighted vertex cover problem onto the independent set $\{X_1, X_4\}$ yields a table as shown on the right-hand side. For example, the entry '7' written against $\{X_1 = 0, X_4 = 1\}$ indicates that when X_1 is prohibited from being in the minimum weighted vertex cover but X_4 is necessarily included in it, then the weight of the minimum weighted vertex cover—$\{X_2, X_3, X_4, X_7\}$ or $\{X_2, X_3, X_4, X_5, X_6\}$ in this case—is equal to 7.

of maximum cardinality. A *vertex cover* is a subset of nodes $U \subseteq V$ such that every edge in E has at least one of its endpoints included in U. A *minimum vertex cover* is a vertex cover of minimum cardinality.

While the problem of computing the maximum matching can be solved using very efficient polynomial-time algorithms (Micali and Vazirani 1980), the problem of computing the minimum vertex cover is NP-hard in general. Nonetheless, for *bipartite graphs*, the minimum vertex cover problem can be solved very efficiently in $O(|V|^{2.5})$ time using a *maxflow* computation (Cormen *et al* 1990). Moreover, even in the general case, the minimum vertex cover can be approximated within a factor of 2 in polynomial time; and this approximation factor can further be improved to $2 - \frac{2}{k}$ for *k-partite graphs* (Hochbaum 1983). It is also well known that the size of a maximum matching serves as a *lower bound* for the size of a minimum vertex cover (Cormen *et al* 1990). Finally, the above results can be extended to the "weighted" case in which the nodes/edges of the graph G are associated with non-negative weights. The *maximum weighted matching* is then defined to be a matching of maximum total weight on its edges, and the *minimum weighted vertex cover* is defined to be a vertex cover of minimum total weight on its nodes.

3 Projections of Minimum Weighted Vertex Cover Problems onto Independent Sets

In this section, we will first introduce the idea of "*projecting*" the minimum weighted vertex cover problem onto an *independent set* of the given graph $G = \langle V, E \rangle$.[3] We will then illustrate and prove a number of interesting properties of these projections. Our study of these projections motivates a special set of transformation techniques—referred to as "*Lifting*"—that we will use to reason about WCSPs by considering each weighted constraint only *locally*.

Consider an undirected graph $G = \langle V, E \rangle$. Let $U = \{u_1, u_2 \ldots u_k\}$ be an independent set of G. We say that a k-bit vector t imposes the following restrictions: (a) the i^{th} bit $t_i = 0$ indicates that the node u_i is necessarily *excluded* from the minimum weighted vertex cover, and (b) the i^{th} bit $t_i = 1$ indicates that the node u_i is necessarily *included* in the minimum weighted vertex cover. The *projection* of the minimum weighted vertex cover problem onto the independent set U is then defined to be a table of size 2^k 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 set to be equal to the weight of the minimum weighted vertex cover *conditioned* on the restrictions imposed by $t^{(j)}$. Figure 1 presents a simple example to illustrate the idea of projecting the minimum weighted vertex cover problem onto an independent set of the given graph.[4]

Given an undirected graph $G = \langle V, E \rangle$ and an independent set $U = \{u_1, u_2 \ldots u_k\}$, let $\mathcal{P}_{G,U}$ denote the projection of the minimum weighted vertex cover problem onto U; and let $\mathcal{P}_{G,U}(t)$ denote the value of the entry corresponding to the k-bit vector t. We will now prove some basic algorithmic properties of the projection $\mathcal{P}_{G,U}$ (see Figures 2 and 3).

Lemma 1: 'COMPUTE-PROJECTION-VALUE' (Figure 2) computes $\mathcal{P}_{G,U}(t)$ for a given k-bit vector t.
Proof: In step 2(a) of the algorithm, we notice that if $t_i = 0$ then the weight of u_i is set to ∞. This ensures the exclusion of u_i from the minimum weighted vertex cover computed in steps 3 and 4. In step 2(b) of the algorithm, we notice that if $t_i = 1$ then u_i is included in the minimum weighted vertex cover (computed in step 4). Further, in this case, all the edges that are incident on u_i are removed from the graph (step 2(b)); this reflects the fact that these edges would now be covered by the inclusion of u_i. The truth of the Lemma then follows simply from the definition of $\mathcal{P}_{G,U}(t)$.

Lemma 2: Procedure 'COMPUTE-MIN-PROJECTION' (Figure 3) computes $\textit{argmin}_t\ \mathcal{P}_{G,U}(t)$ and $\textit{min}_t\ \mathcal{P}_{G,U}(t)$.
Proof: First, we note that the conditions imposed by any k-bit vector t restricts the candidate space for optimization; and therefore, $\mathcal{P}_{G,U}(t) \geq W$. Second, let the assignment returned by the algorithm in Figure 3 be $\hat{t}$. From step 2, $\hat{t}$ is consistent with S on the membership of $u_1, u_2 \ldots u_k$ in the minimum weighted vertex cover; conversely, S is a candidate vertex cover in the space for optimization associated with $\mathcal{P}_{G,U}(\hat{t})$—establishing the condition $\mathcal{P}_{G,U}(\hat{t}) \leq W$. Putting the two results together, we have that for any k-bit vector t, $\mathcal{P}_{G,U}(t) \geq \mathcal{P}_{G,U}(\hat{t})$. This proves that $\hat{t}$ is the required optimal vector of assignments; and clearly, this also proves that $W = \textit{min}_t\ \mathcal{P}_{G,U}(t)$ as required.

We note that both 'COMPUTE-PROJECTION-VALUE' and 'COMPUTE-MIN-PROJECTION' make use of just one call to the minimum weighted vertex cover problem. While

[3] An *independent set* of a graph is a subset of nodes no two of which are connected by an edge.

[4] It is worth noting that the projection is well defined only when U is an independent set. If this is not the case, then there exists some edge (u_{i_1}, u_{i_2}) for $u_{i_1}, u_{i_2} \in U$. The entry corresponding to any k-bit vector that disallows both u_{i_1} and u_{i_2} from being in the minimum weighted vertex cover then becomes undefined because the edge (u_{i_1}, u_{i_2}) cannot be covered in any way.

ALGORITHM: COMPUTE-PROJECTION-VALUE
INPUT: (a) a node-weighted undirected graph $G = \langle V, E \rangle$; (b) an independent set $U = \{u_1, u_2 \ldots u_k\} \subseteq V$; (c) a k-bit vector t.
OUTPUT: the value of the projection $\mathcal{P}_{G,U}(t)$.
(1) $S_1 \leftarrow \{\}$.
(2) For $i = 1, 2 \ldots k$:
  (a) If $t_i = 0$: set the weight of u_i to ∞.
  (b) If $t_i = 1$: $S_1 \leftarrow S_1 \cup \{u_i\}$; remove u_i (and all edges incident on it) from the graph.
(3) Let S_2 be the minimum weighted vertex cover computed for the resulting graph.
(4) Let W be the sum of the weights on all the nodes in $S_1 \cup S_2$.
(5) RETURN: $\mathcal{P}_{G,U}(t) \leftarrow W$.
END ALGORITHM

Figure 2: Shows an algorithm for computing $\mathcal{P}_{G,U}(t)$. The algorithm makes use of one call to the problem of computing the minimum weighted vertex cover.

ALGORITHM: COMPUTE-MIN-PROJECTION
INPUT: (a) a node-weighted undirected graph $G = \langle V, E \rangle$; (b) an independent set $U = \{u_1, u_2 \ldots u_k\} \subseteq V$.
OUTPUT: (a) the optimal t^* such that $t^* = \textit{argmin}_t \, \mathcal{P}_{G,U}(t)$; (b) the optimal value $\mathcal{P}_{G,U}(t^*)$.
(1) Compute the minimum weighted vertex cover on G. Let S be the set of nodes included in this cover, and let W be the total weight of the nodes in S.
(2) For all $u_i \in U$:
  (a) If $u_i \in S$: set $t^*_i \leftarrow 1$.
  (b) If $u_i \notin S$: set $t^*_i \leftarrow 0$.
(3) RETURN: (a) t^*: optimal assignment vector; (b) W: optimal value.
END ALGORITHM

Figure 3: Shows an algorithm for computing the optimal t^* such that $t^* = \textit{argmin}_t \, \mathcal{P}_{G,U}(t)$; the optimal value $\mathcal{P}_{G,U}(t^*)$ is also returned. We note that the algorithm makes use of just one call to the problem of computing the minimum weighted vertex cover.

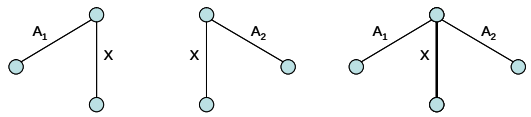


Figure 6: Illustrates the critical problem associated with choosing maximum weighted matchings for providing lifted representations of the weighted constraints. The first two diagrams (possibly coming from trying to represent two different weighted constraints) show the variable X and the auxiliary variables A_1 and A_2 respectively. "Combining" the combinatorial structures by merging the edges that represent X leads to a scenario (as shown in the rightmost diagram) where spurious constraints are introduced between the auxiliary variables; in particular, both A_1 and A_2 are unnecessarily disallowed from being set to 'True' ('1') simultaneously (as the edges representing them now share a common end-point).

this observation follows merely from the definition of a projection for the former algorithm, it is much more interesting in the case of the latter algorithm.

4 Lifted Representations for WCSPs

We will now present important results that relate *projections* to the computational aspects of solving WCSPs. As a first step, we make the simple observation that the result of projecting the minimum weighted vertex cover problem onto an independent set U of the given graph produces a table of size $2^{|U|}$; in some sense, this table can be viewed as a weighted constraint over $|U|$ Boolean variables. Conversely, given a weighted constraint, we can think about designing a "lifted" representation for it so as to be able to view it as the projection of a minimum weighted vertex cover problem in some intelligently constructed node-weighted undirected graph.[5] Later in the paper, we will show how we can build such a lifted representation for any given weighted constraint using a *tripartite graph*. For now, however, we will concentrate only on the computational aspects of solving WCSPs when the lifted representations for each of the weighted constraints are already given to us.

Figure 4 shows an example WCSP over 3 Boolean variables. Here, there are 3 unary weighted constraints and 3 binary weighted constraints; and their lifted representations (as projections of minimum weighted vertex cover problems) are shown next to each of them. Further, the figure also illustrates how a *composite graph* is obtained from the individual graphs corresponding to each of the weighted constraints. In the composite graph, nodes that represent the same variable are simply "merged"—along with their edges—and every "composite" node is given a

[5] This graph can involve other auxiliary nodes.

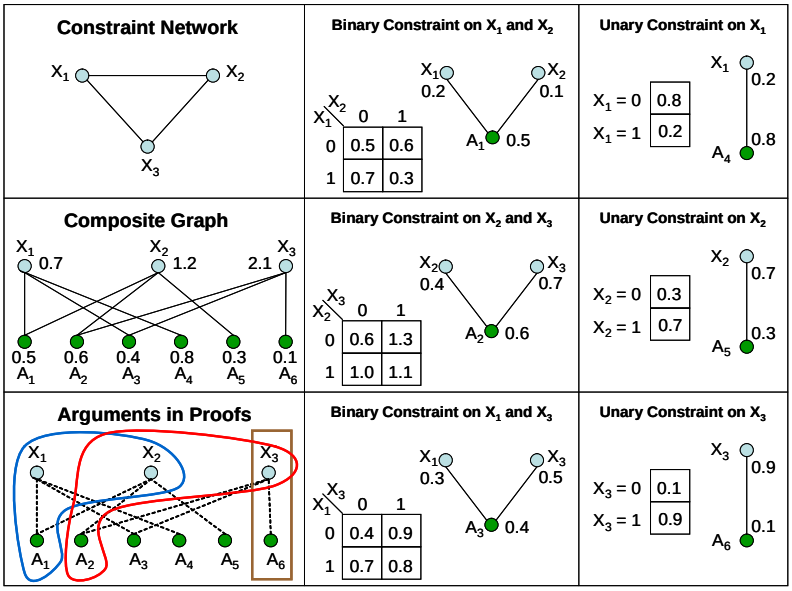


Figure 4: Shows an example WCSP. The 6 unary/binary weighted constraints are shown along with their lifted representations in the rightmost two columns. The composite graph is shown in the 2^{nd} row of the 1^{st} column; and the arguments used in the proof of Lemma 3 are illustrated in the 3^{rd} row of the 1^{st} column. The encircled subgraphs are indicative of the independence of the corresponding subproblems when all the X-variables are instantiated.

weight equal to the sum of the individual weights. Figure 5 presents the procedure for constructing the composite graph; and the following Lemmas prove some very interesting properties of the composite graph in the general case.[6]

Lemma 3: Consider a complete assignment q (i.e., an assignment of values to all the variables from their respective domains). The cost of q can be computed simply by running 'COMPUTE-PROJECTION-VALUE' on the composite graph.

Proof: The cost of q is given by the sum of the costs defined locally by each weighted constraint. From Lemma 1, the cost defined locally by C_i can be computed by running 'COMPUTE-PROJECTION-VALUE' on H_i (see procedure in Figure 5). Therefore, it suffices for us to prove that running 'COMPUTE-PROJECTION-VALUE' on the composite graph is equivalent to running it on each of the individual graphs $H_1, H_2 \ldots H_M$ and summing the results. Consider the total weight contributed by the X-nodes—say, X_r $(1 \leq r \leq N)$ in particular. When $X_r = 0$, the total weight contributed by X_r in any H_i is 0, and this is also the case in the composite graph. When $X_r = 1$, the total weight contributed by X_r is equal to the sum of the weights associated with it in each of the individual graphs that it appears in. By construction (step 2(a)(B) in Figure 5), this total weight is equal to the weight contributed by X_i in the composite graph. Now consider the total weight contributed by the auxiliary nodes. It is easy to see that once the nodes $X_1, X_2 \ldots X_N$ are instantiated in the composite graph, the optimal values for the auxiliary variables coming from one graph are *independent* of the optimal values for the auxiliary variables coming from any other graph; and this establishes that any auxiliary variable—say, coming from the graph H_j—is chosen to be in the minimum weighted vertex cover of the composite graph if and only if it is chosen to be in the minimum weighted vertex cover of H_j. Therefore, the total weight contributed by the auxiliary nodes also remains the same in the composite graph—hence proving the Lemma.

Lemma 4: The optimal (minimum) cost complete assignment q^* (for the given WCSP) can be computed simply by running the procedure 'COMPUTE-MIN-PROJECTION' on the composite graph.

Proof: From Lemma 2, the assignment returned by running the procedure 'COMPUTE-MIN-PROJECTION' (on the composite graph) is optimal with respect to the composite graph. From Lemma 3, the cost of any complete assignment can be computed from the composite graph. Put together, the returned assignment is optimal for the given WCSP—hence proving the Lemma.

It is worth noting that the arguments used in the proofs of the above Lemmas are somewhat similar to those used in *loop-cutset conditioning* (Pearl 1986). It is the above property of the *vertex cover* problem that makes it an intelligently chosen combinatorial problem useful for building lifted representations of the weighted constraints. Another combinatorial structure that exhibits this property is the *maximum weighted independent set*. On the other hand, the *maximum weighted matching* problem (where the val-

[6]These Lemmas allow us to reason about each weighted constraint only *locally*, and this special reduction mechanism is therefore given the name "*Lifting*".

ALGORITHM: BUILD-COMPOSITE-GRAPH
INPUT: (a) a WCSP with variables $X_1, X_2 \ldots X_N$ and weighted constraints $C_1, C_2 \ldots C_M$; (b) lifted graphical representations $H_1, H_2 \ldots H_M$ for each of the weighted constraints—the graph H_i corresponds to the weighted constraint C_i.
OUTPUT: a *composite graph* that provides a lifted representation for the entire WCSP.
(1) For $i = 1, 2 \ldots M$:
  (a) Give the auxiliary variables in H_i unique names.
(2) For $i = 1, 2 \ldots N$:
  (a) "Merge" all copies of X_i by doing the following:
    (A) If X_i has an edge to an auxiliary node A in any of the graphs $H_1, H_2 \ldots H_M$, then introduce an edge between the "merged" copy of X_i and A in the composite graph as well.
    (B) Set the weight on the "merged" copy of X_i to be equal to the sum of the weights assigned to it in each of the individual graphs $H_1, H_2 \ldots H_M$ that it appears in.
(3) RETURN: the resulting *composite graph*.
END ALGORITHM

Figure 5: A straightforward procedure for building the composite graph from the individual graphs that represent each of the weighted constraints in a WCSP. The composite graph provides a lifted representation for the entire WCSP.

ues of the Boolean variables in the given WCSP are represented using the presence/absence of certain edges in the maximum weighted matching) may be used to represent interesting weighted constraints *locally*, but as Figure 6 shows, the "combination" of the representations built for different weighted constraints introduces spurious dependencies between the auxiliary variables, and therefore does not suit our purposes.

5 Computational Results for WCSPs

We will now prove an important Theorem and illustrate the power of this Theorem in identifying several interesting tractable classes of WCSPs. We will also discuss the computational aspects of solving WCSPs in the general case, and examine the implications of our approach when combined with other mathematical tools.

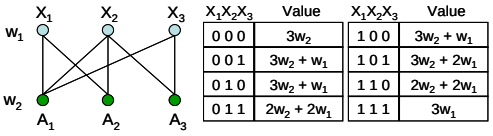
Theorem 5: The language $\mathcal{L}_{\text{bipartite}}$ of all weighted constraints that have *lifted bipartite graph representations* is tractable.[7]

Proof: From algorithm 'BUILD-COMPOSITE-GRAPH' in Figure 5, it is clear that when every weighted constraint in a WCSP has a lifted bipartite graph representation with the X-variables belonging to the same partition, then the composite graph is also bipartite with all the X-variables belonging to the same partition. The truth of the Theorem then follows simply from the fact that in any bipartite graph, the minimum weighted vertex cover problem can be solved in polynomial time (Cormen *et al* 1990).

5.1 Boolean Variables and Binary/Non-Binary Constraints

We first consider WCSPs with Boolean variables and binary constraints. Even in this simple case, the kinds of problems that we can speak about significantly differ in their

[7] Of course, all the X-variables in any graph are required to have the same color—i.e., belong to the same partition.



$X_1X_2X_3$	Value	$X_1X_2X_3$	Value
0 0 0	$3w_2$	1 0 0	$3w_2 + w_1$
0 0 1	$3w_2 + w_1$	1 0 1	$3w_2 + 2w_1$
0 1 0	$3w_2 + w_1$	1 1 0	$2w_2 + 2w_1$
0 1 1	$2w_2 + 2w_1$	1 1 1	$3w_1$

Figure 8: Shows a bipartite graph representing a weighted constraint over the 3 Boolean variables X_1, X_2 and X_3. A_1, A_2 and A_3 are the auxiliary variables.

associated tractability results. For example, both the *min-st-cut* problem and the *max-cut* problem can be encoded as WCSPs with Boolean variables and binary constraints;[8] but while the former problem can be solved in polynomial time, the latter problem is NP-hard. Figure 7 sheds some light on such WCSPs; in particular, it shows that: (a) any Boolean unary weighted constraint has a simple lifted bipartite graph representation; (b) the *min-st-cut* constraints are particular cases of weighted constraints that have a simple lifted bipartite graph representation as a V-structure; and (c) the *max-cut* constraints are particular cases of weighted constraints that have a simple lifted representation as a U-structure (that is not bipartite).[9] The following important conclusions can be drawn immediately: (a) a generalization of the *min-st-cut* problem with arbitrary unary weighted constraints is tractable;[10] (b) the entire space of weighted constraints resulting from varying the parameters w_1, w_2 and w_3 (in the V-structure) is tractable; and (c) the absence of a

[8] For the *min-st-cut* problem, unary weighted constraints on X_s and X_t ensure that they are assigned the values 0 and 1 respectively; and for every edge $\langle v_i, v_j \rangle$ in the graph, a binary weighted constraint between X_i and X_j yields a value of 1 when $X_i \neq X_j$, and 0 otherwise. For the *max-cut* problem, the binary weighted constraints are reversed.

[9] Note that the X-variables have to be in the same partition.

[10] Similar problems were identified as being tractable in (Kolmogorov and Zabih 2004) using different combinatorial arguments.

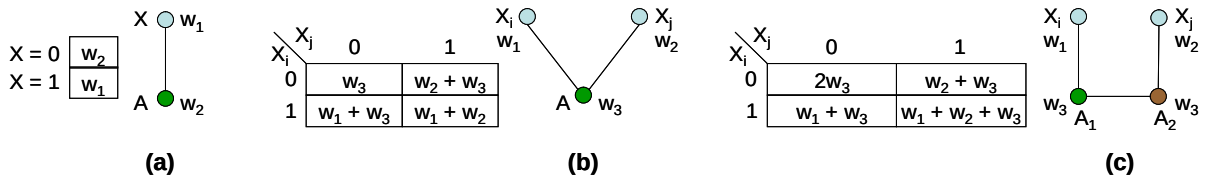


Figure 7: (a) shows that any Boolean unary weighted constraint has a trivial bipartite graph representation; (b) shows the bipartite graph representation (V-structure) for generalizations of the *min-st-cut* constraints; (c) shows the tripartite graph representation (U-structure) for generalizations of the *max-cut* constraints. The *min-st-cut* and *max-cut* constraints in (b) and (c) respectively become apparent when $w_1 = w_2 = w_3/2$ (and the additive constants are factored out).

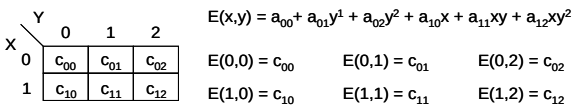


Figure 10: Illustrates how a weighted constraint can be represented as a multivariate polynomial. $E(X, Y)$ is the required polynomial; and its coefficients can be computed by solving the system of 6 linear equations with 6 unknowns.

lifted bipartite graph representation for *max-cut* constraints is consistent with its intractability.

As a next step, we present a simple example in Figure 8 to illustrate how we can generalize our techniques to reason about non-binary weighted constraints. The mere existence of the lifted bipartite graph representation establishes the tractability of the kinds of ternary weighted constraints shown in the figure. Further, setting different values for w_1 and w_2 yields different kinds of tractable (convex) functions. In general, several parameters in the bipartite graphs can be adjusted to yield a multitude of tractable classes of WCSPs. These include: (a) the *weights* on the nodes, (b) the *graphical structures* of the bipartite graphs, and (c) the *encoding mechanism* between the values of individual variables and the presence/absence of certain nodes in the minimum weighted vertex covers.

5.2 Higher Domain Sizes and Constraint Arities

We begin this subsection by proving an interesting result relevant to more general scenarios where variables can take values from the set $\{0, 1 \ldots K\}$. (K is allowed to be different for different variables.) We show that it is possible to efficiently solve the *minimization* problem over these variables for any objective function that can be expressed as a *bounded-degree multivariate polynomial with the positive coefficients being restricted to terms of degree* ≤ 1.[11]

Figure 9 illustrates how to construct the bipartite graphs equivalent to any of the terms in the multivariate polynomial (of the above-mentioned kind). We use K nodes to represent the value of a variable with domain $\{0, 1 \ldots K\}$; and we use the convention that the value of this variable is equal to the number of nodes (amongst these K nodes) that are present in the minimum weighted vertex cover. The leftmost diagram in Figure 9 shows that any linear term $w \cdot X$ (w may be +ve or -ve) has a simple bipartite graph representation. The middle diagram in Figure 9 illustrates the more interesting *cross-product* construction of a bipartite graph for a given -ve term. Consider a term $-w \cdot (X \cdot Y \cdot Z)$ (where $w \geq 0$). Suppose that the domain sizes of X, Y and Z are 4, 4 and 3 respectively; we would have 3 nodes representing the value of X, 3 nodes for Y, and 2 nodes for Z. It is easy to see that if the values assigned to X, Y and Z are $0 \leq k_1 \leq 3$, $0 \leq k_2 \leq 3$ and $0 \leq k_3 \leq 2$ respectively, then the size of the minimum weighted vertex cover is $w \cdot (18 - k_1 k_2 k_3) + k_1 + k_2 + k_3$. Factoring out the additive constants and treating the linear terms as shown before, the bipartite graph (in the middle diagram of Figure 9) essentially represents the term $-w \cdot X \cdot Y \cdot Z$ as required. Similar arguments are used to establish the validity of the *cross-product* construction for any given -ve term in the multivariate polynomial.

5.3 Tools: Change of Variables and Taylor Series

We will now briefly comment on a few more implications of the foregoing discussions. First, we note that a simple graph-theoretic trick allows us to substitute $(|D_i| - 1 - X_i)$ for X_i; here $|D_i|$ is the domain size of X_i. The rightmost diagram in Figure 9 illustrates how this is done for an example variable Y by introducing an intermediate level of nodes with large weights on them. We also note that although this technique—in conjunction with the *cross-product* construction—allows us to create +ve nonlinear terms in a multivariate polynomial,[12] the graph is no more bipartite; instead, it becomes tripartite (as shown in the figure). The only case when the graph continues to be bipartite is when all the participating variables in the constraint undergo this transformation. Such a case yields terms of the kind $-w \cdot (3 - X) \cdot (3 - Y) \cdot (2 - Z)$ which are still tractable for their bipartite graph representations. Moreover, these nonlinear terms are monotonically *increasing* with respect to the variables—unlike the monotonically *decreasing* terms

[11] This is equivalent to dealing with interesting real-life situations that pose linear "biases" on the values of individual variables in addition to potential functions/weighted constraints (of bounded arities) that prefer the participating variables to have higher values.

[12] Lower degree terms are cancelled recursively.

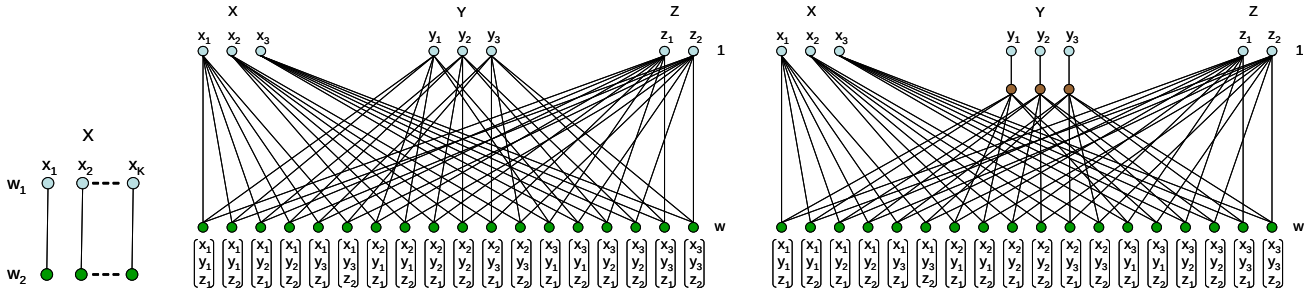


Figure 9: The leftmost diagram shows the bipartite graph equivalent to $w \cdot X$; here, we set w_1 and w_2 so that $w_1 - w_2 = w$. The middle diagram illustrates the *cross-product* construction of the bipartite graph for the term $-w \cdot (X \cdot Y \cdot Z)$; here, $w \geq 0$. The rightmost diagram illustrates the construction of the tripartite graph for the term $w \cdot (X \cdot Y \cdot Z)$; here, $w \geq 0$.

of the form $-w \cdot X \cdot Y \cdot Z$ (which were previously shown to have bipartite graph representations). A proper blend of these monotonically increasing and decreasing terms allows us to construct even richer classes of tractable functions.

We also remark that in many discrete combinatorial optimization problems, the objective function involves analytic functions of various kinds. The *Taylor* series expansions of such functions relates well to our foregoing discussion of (multivariate) polynomials. A variety of analytic functions (e.g. hyperbolic functions like $-sinh\,(3X + 4Y)$) have only -ve nonlinear terms in their *Taylor* series expansions, and can therefore be approximated well by tractable polynomials with bipartite graph representations. Further, the "change of variable" method enriches the class of analytic functions that are amenable to these bipartite graph representations.

5.4 Solving a Given Instance of the WCSP

We will now illustrate how any weighted constraint can be represented as a multivariate polynomial.[13] Consider the example binary weighted constraint in Figure 10. The constraint can be encoded as a multivariate polynomial of degree 1 in X and degree 2 in Y.[14] The coefficients of the polynomial can be computed by using a standard *Gaussian Elimination* procedure for solving systems of linear equations. The linear equations arise from substituting different values to the variables X and Y, and equating the results to the corresponding entries in the (weighted) constraint. We also observe that the number of terms in the multivariate polynomial is equal to the size of the constraint; and the size of the *cross-product* construction (for the terms in this multivariate polynomial) is only polynomial in the size of the weighted constraint.

Theorem 6: Any given WCSP can be reduced to the minimum weighted vertex cover problem in a tripartite graph; and the size of this tripartite graph is only polynomial in the size of the WCSP.

Proof: We know that any weighted constraint can be cast as a multivariate polynomial; further, the -ve/+ve terms in this polynomial can be given lifted representations as bipartite/tripartite graphs (as shown in Figure 9). Now, similar to the arguments used in Theorem 5, when every weighted constraint in a WCSP has a lifted bipartite/tripartite graph representation with the X-variables belonging to the same partition, then the composite graph is tripartite with all the X-variables belonging to the same partition. The truth of the Theorem then follows from the observation made above.

We can now see that the complexity of solving a given instance of the WCSP is exponential only in the size of the *smallest* partition—in terms of the number of nodes—of the tripartite graph constructed for it. This is because the minimum weighted vertex cover problem can be solved in polynomial time for a bipartite graph; and every possible combination of decisions to include/exclude the nodes of the smallest partition in the vertex cover can be evaluated to find the optimal one. We note that one of these partitions consists of the original N variables—leading us to the obvious upper bound of characterizing the problem to be exponential in N. However, this partition may not be the smallest—in which case, our framework yields a much tighter characterization; in particular, when there is sufficient *numerical* structure in the weighted constraints, the composite graph is only bipartite, and such WCSPs can be solved in polynomial time. Even when the composite graph is not bipartite, our framework allows us to computationally leverage the *numerical* structure of the weighted constraints—when, for example, they look more like the polynomial-time solvable *min-st-cut* constraints than the NP-hard *max-cut* constraints.

6 Related Work

The works of several researchers in the AI/Theory communities are related to the work presented in this paper. First, a lot of recent work in AI has been motivated by the problems that relate to detecting "hidden" variables, determining their relationship to other variables, etc.[15] Second, recent works in the Theory community report on many *lift-and-project* methods for constructing projection-based represen-

[13] This is a common technique in coding/complexity theory.

[14] In general, if the domain of a variable is $\{0, 1 \ldots K\}$, then the polynomial is of degree K in this variable.

[15] We note that in our approach, we are primarily concerned with the *numerical* structure of the weighted constraints/potentials rather than the structure of the *variable-interaction* graph.

tations of general 0-1 polytopes. These include the procedures of Sherali-Adams, Lovász-Schrijver and Lasserre (Lasserre 2001). These methods involve a sequence of Linear Programming relaxations, and in the worst case, could require lifting the original problem to a space with an exponential number of dimensions. Many combinatorial problems have been studied for the presence of additional structure that may obviate the exponential number of dimensions; an efficient approximation algorithm for the *max-cut* problem, for example, uses only a polynomial number of additional dimensions. For the kinds of problems that we dealt with in this paper, however, the lifting techniques that we proposed are more direct and relevant. Further, interesting connections to graph-theoretical results are also manifested in our approach.

The works of several other researchers also relate to more specific details of the work presented in this paper. First, the arguments underlying our lifting techniques are akin to those used in *loop-cutset conditioning* for Bayesian network inference (Pearl 1986) and/or graph-based search strategies for solving CSPs (Dechter 1992). Second, several interesting algorithms have been reported for efficiently solving EMPs on certain kinds of MRFs. For example, the reduction of EMPs on *convex* MRFs to instances of the *min-st-cut* problem is reported in (Ishikawa 2003). Several other related recent advances are mentioned in (Kolmogorov 2005). Another recent development is the *tree-reweighted max-product message passing (TRW)* algorithm (Wainwright *et al* 2003). TRW procedures are inspired by the problem of maximizing a lower bound on the Energy; however, as in the case of ordinary *belief propagation* procedures, they do not always converge (Wainwright *et al* 2003).

7 Conclusions and Future Work

We identified rich tractable classes of WCSPs based on a special set of transformation techniques referred to as "*Lifting*". We showed that WCSPs are reducible to minimum weighted vertex cover problems in tripartite graphs; and many tractable classes of WCSPs can be recognized by their reducibility to minimum weighted vertex cover problems in bipartite graphs. We examined the implications of our approach when combined with other mathematical tools, and provided a framework for tightly characterizing the complexity of solving a given instance of the WCSP (using our approach). Several lines of thought are of interest to us for our future work. Some of these are: (1) a thorough understanding of the implications of our approach on the *approximability* of different kinds of WCSPs, (2) an evaluation of the *lower bounds* generated by computing the maximum weighted matchings on the composite graphs, and (3) the idea of constructing bipartite graphs the minimum weighted vertex covers of which can best "fit" the weighted constraints/potentials of a given WCSP/EMP (so that subsequent combinatorial tasks on it can be done efficiently).

References

Bistarelli S., Fargier H., Montanari U., Rossi F., Schiex T. and Verfaillie G. (1996). Semiring-Based CSPs and Valued CSPs: Basic Properties and Comparison. *Over-Constrained Systems*.

Boutilier C., Brafman R., Domshlak C., Hoos H. and Poole D. (2004). CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *J. Artif. Intell. Res. (JAIR) 21: 135-191*.

Cormen T., Leiserson C. and Rivest R. (1990). *Introduction to Algorithms*. MIT Press, Cambridge, MA.

Dechter R. (1992). Constraint Networks (Survey). *Encyclopedia of Artificial Intelligence*.

Do M., Benton J., van den Briel M. and Kambhampati S. (2007). Planning with Goal Utility Dependencies. *IJCAI'2007*.

Hochbaum D. (1983). Efficient Bounds for the Stable Set, Vertex Cover and Set Packing Problems. *Disc. Appl. Math. 6 (1983)*.

Ishikawa H. (2003). Exact Optimization for Markov Random Fields with Convex Priors. *Transactions on Pattern Analysis and Machine Intelligence 25(10):1333-1336*.

Kolmogorov V. (2005). Primal-Dual Algorithm for Convex Markov Random Fields. *Microsoft Tech. Rep. MSR-TR-2005-117*.

Kolmogorov V. and Zabih R. (2004). What Energy Functions can be Minimized via Graph Cuts? *Transactions on Pattern Analysis and Machine Intelligence 26(2):147-159*.

Lasserre J. (2001). An Explicit Equivalent Positive Semidefinite Program for Nonlinear 0-1 Programs. *SIAM Journal on Optimization, 12:756-769*.

Micali S. and Vazirani V. (1980). An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs. *FOCS'1980*.

Pearl J. (1986). Fusion, Propagation and Structuring in Belief Networks. *Artificial Intelligence, Vol. 29, No. 3, 241-288*.

Schiex T., Fargier H. and Verfaillie G. (1995). Valued Constraint Satisfaction Problems: Hard and Easy Problems. *IJCAI'1995*.

Wainwright M., Jaakkola T. and Willsky A. (2003). MAP Estimation via Agreement on (Hyper)Trees: Message-Passing and Linear-Programming Approaches. *Technical Report, UC Berkeley, CS Division*.