Submodular Constraints and
Planar Constraint Networks: New Results

T. K. Satish Kumar[*]
Department of Computer Science
University of Southern California
tkskwork@gmail.com

Liron Cohen
Department of Computer Science
University of Southern California
lironcoh@usc.edu

Sven Koenig
Department of Computer Science
University of Southern California
skoenig@usc.edu

In this paper, we present fast polynomial-time algorithms for solving classes of submodular constraints over Boolean domains. We pose the identified classes of problems within the general framework of Weighted Constraint Satisfaction Problems (WCSPs), reformulated as minimum weighted vertex cover problems. We examine the Constraint Composite Graphs (CCGs) associated with these WCSPs and provide simple arguments for establishing their tractability. We construct simple - almost trivial - bipartite graph representations for submodular cost functions, and reformulate these WCSPs as max-flow problems on bipartite graphs. By doing this, we achieve better time complexities than state-of-the-art algorithms. We also use CCGs to exploit planarity in variable interaction graphs, and provide algorithms with significantly improved time complexities for classes of submodular constraints. Moreover, our framework for exploiting planarity is not limited to submodular constraints. Our work confirms the usefulness of studying CCGs associated with combinatorial problems modeled as WCSPs.

Introduction

In many application domains, we are required to efficiently represent and reason about factors like fuzziness, probabilities, preferences, and/or costs. Many extensions to the basic framework of Constraint Satisfaction Problems (CSPs) (Dechter 2003) have been introduced to incorporate and reason about such "soft" constraints. These include variants like *fuzzy-CSPs*, *probabilistic-CSPs*, and Weighted-CSPs (WCSPs). A WCSP is an *optimization* version of a CSP in which the constraints are no longer "hard," but are extended by associating non-negative *costs* with the tuples. The goal is to find an assignment of values to all variables from their respective domains such 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* on subsets of the variables. Each variable X_i is associated with a discrete-valued *domain* $D_i \in \mathcal{D}$, and each 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 every possible combination of values to the variables in S_i. The *arity* of the constraint C_i is equal to $|S_i|$. An *optimal* solution is an assignment of values to all variables (from their respective domains) so that the *sum* of the costs (as specified locally by each weighted constraint) is *minimized*. In a Boolean WCSP, the size of any variable's domain is equal to 2, that is, $D_i = \{0, 1\}$ for all $i \in \{1, 2 \ldots N\}$. Boolean WCSPs are representationally as powerful as WCSPs; and it is well known that optimally solving Boolean WCSPs is NP-hard in general (Dechter 2003). The *constraint network* (also called the *constraint graph* or the *variable interaction graph*) associated with a WCSP instance is an undirected graph where a node represents a variable and an edge (X_i, X_j) exists if and only if X_i and X_j appear together in some constraint.

Boolean WCSPs can be used to model important combinatorial problems arising in different application domains. Examples include (but are not limited to) representing and reasoning about user preferences (Boutilier et al. 2004), over-subscription planning with goal preferences (Do et al. 2007), combinatorial auctions (Sandholm 2002), and bioinformatics (Sanchez, de Givry, and Schiex 2007). They also arise as *Energy Minimization Problems* in probabilistic settings. In computer vision applications, for example, tasks such as image restoration, total variation minimization, and panoramic image stitching can be formulated as Boolean WCSPs derived in the context of *Markov Random Fields* (Kolmogorov 2005). In addition, many real-world domains exhibit *submodularity* in the cost structure and *planarity* in the variable interaction graphs, that are worth exploiting for computational benefits.

Submodular cost functions are characterized by a natural "diminishing returns" property that makes them useful in game theory, sensor placement, and semantic segmentation of images, among others. Submodular constraints over Boolean domains correspond directly to *submodular set functions*. A set function $\psi : 2^V \to \mathbb{Q}$ defined on all subsets of a set V is *submodular* if and only if, for all subsets $S, T \subseteq V$, we have $\psi(S \cup T) + \psi(S \cap T) \leq \psi(S) + \psi(T)$. A submodular constraint is a weighted constraint with a submodular cost function. Here, the correspondence is in light of the observation that any subset S can be interpreted as specifying the Boolean variables in V that are set to 1. Boolean WCSPs with submodular constraints are known to

---

[*]Alias: Satish Kumar Thittamaranahalli

be tractable (Zivny and Jeavons 2008). However, the general algorithm for solving Boolean WCSPs with submodular constraints has a time complexity of $O(N^6)$, which is not very practical. Specific classes of submodular constraints have been shown to be related to graph cuts, and are therefore solvable more efficiently (Zivny and Jeavons 2008).

By definition, *planar graphs* are those that can be drawn on a planar surface without any two edges crossing each other. Many combinatorial problems are known to be easier to solve on planar graphs (Baker 1994). Planar graphs, however, do not necessarily have a bounded *treewidth*.[1] This means that planar graphs exhibit additional computational properties that are not necessarily captured by a treewidth-based characterization. Very little work has been done on exploiting planarity in constraint networks associated with (Boolean) WCSPs even though planarity occurs naturally in real-world domains. For example, 2-dimensional pictures in computer vision, 2-dimensional surfaces in sensor placement, and 2-dimensional fields for circuit layouts and transportation networks are all indicative of the potential for exploiting planarity towards computational benefits.

Consider the problem of defending a perimeter with identical agents that are used for surveillance. The problem of choosing vantage points from a given set of possible locations is much like the problem of sensor placement. The latter problem can be modeled as a Boolean WCSP with the objective of maximizing mutual information between chosen and unchosen locations. Here, the weighted constraints can be designed to be submodular (Krause, Singh, and Guestrin 2008). Further, planarity is also commonplace in such spatial reasoning problems.

Constraint Composite Graphs (CCGs) are combinatorial structures associated with optimization problems posed as WCSPs. They provide a unifying framework for exploiting both the *graphical* structure of the variable interactions as well as the *numerical* structure of the weighted constraints (Kumar 2008a). Moreover, establishing tractability results for various subclasses of WCSPs is often much simpler when using CCGs. An important example is proving tractability of the language $\mathcal{L}_{bipartite}$ (Kumar 2008b). In this paper, we discuss both submodularity and planarity in the light of CCGs by reformulating WCSPs as *minimum weighted vertex cover* problems. By doing so, we are able to: (1) construct simple bipartite graph representations for important classes of submodular constraints, thereby translating them into max-flow problems on bipartite graphs; (2) identify tractable classes of WCSPs that have only a logarithmic number of constraints not included in $\mathcal{L}_{bipartite}^{Boolean}$ ($\mathcal{L}_{bipartite}$ for Boolean variables); and (3) exploit planarity in variable interaction graphs to design algorithms with significantly improved time complexities for various classes of WCSPs. In general, our work confirms the usefulness of studying the CCGs associated with combinatorial problems modeled as WCSPs.

---

[1] The treewidth is a measure of the size of the largest subproblem that needs to be solved in a dynamic programming-based approach for solving the original problem.
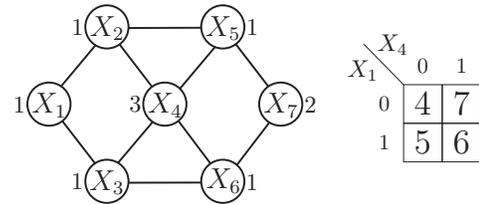

Figure 1: The table on the right-hand side represents the projection of the minimum weighted VC problem onto the IS $\{X_1, X_4\}$ of the node-weighted undirected graph on the left-hand side. (The weights on X_4 and X_7 are set to 3 and 2, respectively, while all other nodes have unit weights.) The entry '7' in the cell $(X_1 = 0, X_4 = 1)$, for example, indicates that, when X_1 is prohibited from being in the minimum weighted VC but X_4 is necessarily included in it, then the weight of the minimum weighted VC - $\{X_2, X_3, X_4, X_7\}$ or $\{X_2, X_3, X_4, X_5, X_6\}$ - is 7.

Background on CCGs

In an undirected graph $G = \langle V, E \rangle$, $U = \{u_1, u_2 \ldots u_k\}$ is said to be an *independent set* (IS) of G if and only if no two nodes in U are connected by an edge in E. A *vertex cover* (VC) is a set of nodes $S \subseteq V$ such that every edge has at least one end point in S. A *minimum* VC is a VC of minimum size. When non-negative weights are associated with nodes, the *minimum weighted* VC is defined to be a VC of minimum weight.

The concept of the minimum weighted VC on a given undirected graph $G = \langle V, E \rangle$ can be *extended* to the notion of projecting minimum weighted VCs onto a given IS $U \subseteq V$. The input to such a projection is the graph G as well as an identified IS $U = \{u_1, u_2 \ldots u_k\}$. The output is a *table* of 2^k numbers. Each entry in this table corresponds to a k-bit vector. We say that a k-bit vector t imposes the following restrictions: (a) if the ith bit t_i is 0, the node u_i is necessarily excluded from the minimum weighted VC; and (b) if the ith bit t_i is 1, the node u_i is necessarily included in the minimum weighted VC. The projection of the minimum weighted VC problem onto the IS U is then defined to be a table with entries corresponding to each of the 2^k possible k-bit vectors $t^{(1)}, t^{(2)} \ldots t^{(2^k)}$. The value of the entry corresponding to $t^{(j)}$ is equal to the weight of the minimum weighted VC *conditioned* on the restrictions imposed by $t^{(j)}$. Figure 1 presents a simple example to illustrate the notion of projecting minimum weighted VC problems onto an IS in a node-weighted undirected graph.

The table of numbers produced above can be viewed as a weighted constraint over $|U|$ Boolean variables. Conversely, given a (Boolean) weighted constraint, we can think about designing a "lifted" representation for it so as to be able to view it as the projection of a minimum weighted VC problem in some intelligently constructed node-weighted undirected graph. This idea was first discussed in (Kumar 2008b). The benefit of constructing these graphical representations for individual constraints lies in the fact that the "lifted" graphical representation for the entire WCSP can be

Constraint Graph

Binary Constraints

$X_1 X_2$	
0 0	0.5
0 1	0.6
1 0	0.7
1 1	0.3

0.2 (X_1) 0.1 (X_2) 0.5 (A_1)

$X_2 X_3$	
0 0	0.6
0 1	1.3
1 0	1.0
1 1	1.1

0.4 (X_2) 0.7 (X_3) 0.6 (A_2)

$X_1 X_3$	
0 0	0.4
0 1	0.9
1 0	0.7
1 1	0.8

0.3 (X_1) 0.5 (X_3) 0.4 (A_3)

Unary Constraints

X_1	
0	0.8
1	0.2

0.2 (X_1) 0.8 (A_4)

X_2	
0	0.3
1	0.7

0.7 (X_2) 0.3 (A_5)

X_3	
0	0.1
1	0.9

0.9 (X_3) 0.1 (A_6)

Constraint Composite Graph

0.7 (X_1) 1.2 (X_2) 2.1 (X_3)

0.5 (A_1) 0.6 (A_2) 0.4 (A_3) 0.8 (A_4) 0.3 (A_5) 0.1 (A_6)
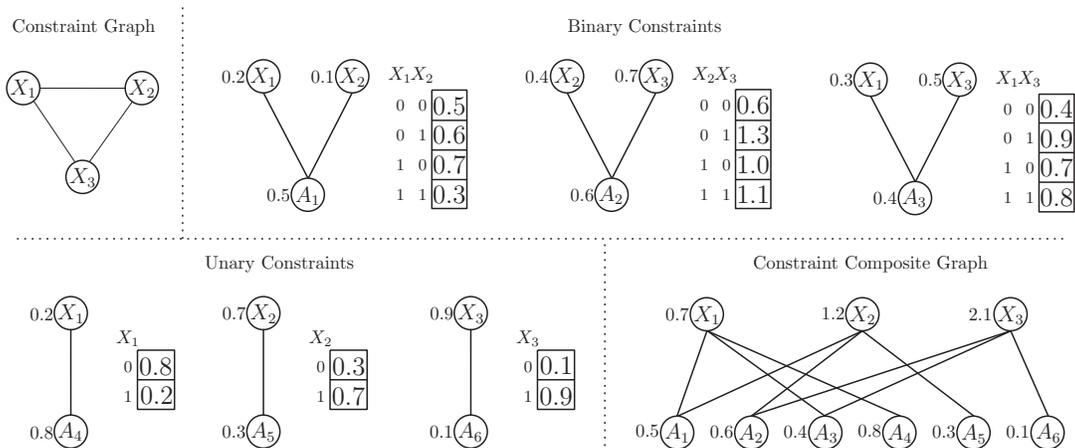
Figure 2: Shows a WCSP over 3 Boolean variables. The constraint network is shown in the top-left cell, and the 6 binary and unary weighted constraints are shown along with their lifted graphical representations in the 1st and 2nd rows. The CCG is shown in the bottom-right cell.

obtained simply by "merging" them. This "merged" graph is referred to as the CCG associated with the WCSP.

Figure 2 shows an example WCSP over 3 Boolean variables to illustrate the construction of the CCG. Here, there are 3 unary weighted constraints and 3 binary weighted constraints; and their lifted representations (as projections of minimum weighted VC problems) are shown next to them. The figure also illustrates how the CCG is obtained from the individual graphs representing each of the weighted constraints. In the CCG, nodes that represent the same variable are simply "merged" - along with their edges - and every "composite" node is given a weight equal to the sum of the individual weights of the merged nodes. Computing the minimum weighted VC for the CCG yields a solution for the WCSP; namely, if X_i is in the minimum weighted VC, then it is assigned the value 1 in the WCSP, else it is assigned the value 0 in the WCSP.

Any given weighted constraint on Boolean variables (that is, a Boolean weighted constraint) can be represented graphically using a *tripartite* graph, which can be constructed in polynomial time (Kumar 2008a). In many cases, the lifted graphical representations even turn out to be only *bipartite*. Since the resulting CCG is also bipartite if each of the individual graphical representations are bipartite, the tractability of the language $\mathcal{L}_{bipartite}^{Boolean}$ - the language of all Boolean weighted constraints with a bipartite graphical representation - is readily established. This is because solving minimum weighted VC problems on bipartite graphs is reducible to max-flow problems, and can therefore be solved efficiently in polynomial time.

Lifted Graphical Representations for Weighted Constraints

Boolean weighted constraints can be represented as multivariate polynomials on the variables participating in that constraint (Zivny and Jeavons 2008; Kumar 2008a). The coefficients of the polynomial can be computed with a standard

Gaussian Elimination procedure for solving systems of linear equations. The linear equations themselves arise from substituting different combinations of values to the variables, and equating them to the corresponding entries in the weighted constraint. One way to build a graphical representation for a given weighted constraint is therefore to simply: (a) build the graphical representations for each of the individual terms in the multivariate polynomial; and (b) "merge" these individual graphical representations (Kumar 2008a).

We need to construct graphical representations for only three kinds of terms: (1) linear terms, (2) negative nonlinear terms, and (3) positive nonlinear terms. First, consider building a graphical representation for a given (positive or negative) linear term. Any such term can be represented by a single edge that connects the corresponding variable to an auxiliary node. The non-negative weights on the two nodes are set appropriately as shown in Figure 3(a).[2]

Now, consider building a graphical representation for a given negative nonlinear term, say, $-w \cdot (X_i \cdot X_j \cdot X_k)$ for $w > 0$. A simple "flower" structure, as shown in Figures 3(b)&(c), serves the requirements. The "flower" structure makes use of one auxiliary node that is connected to all variables appearing in the term. A unit weight is assigned to all original variables while a weight of w is assigned to the auxiliary node. Since the only case in which the auxiliary node is excluded from the minimum weighted VC is when all original variables are set to 1, the weight of the minimum weighted VC for the example in Figure 3(c) is $X_i + X_j + X_k + w - w \cdot (X_i \cdot X_j \cdot X_k)$. After canceling the linear lower-order terms by constructing graphical representations of their negatives (as discussed above), we obtain a graphical representation for the term $-w \cdot (X_i \cdot X_j \cdot X_k)$ as required. Once again, the additive constant can be ignored.

Finally, consider building a graphical representation for a given positive nonlinear term. Unlike negative nonlinear terms, positive nonlinear terms do not always have bipartite

---

[2]Additive constants do not change the optimal solution.

$(w_1 - w_2)X_i$ $-w(X_iX_j)$ $-w(X_iX_jX_k)$ $w(X_iX_jX_k)$ $-w(1-X_i)(1-X_j)(1-X_k)$
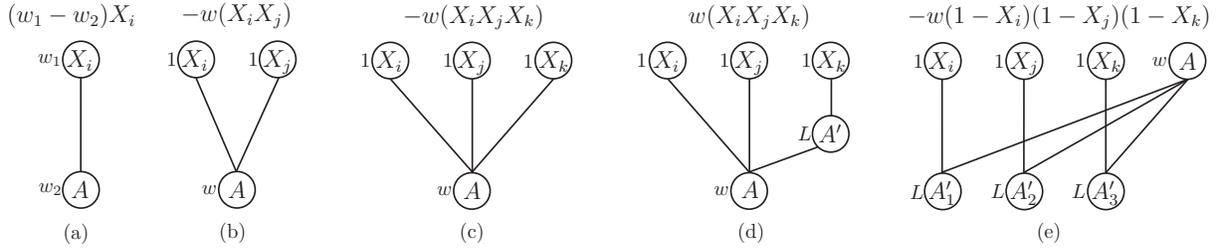
(a) (b) (c) (d) (e)

Figure 3: Lifted graphical representations for different kind of terms. (a) represents a linear term, either positive or negative, where w_1 and w_2 are chosen appropriately. (b) represents a negative quadratic term. (c) represents a negative cubic term. (d) illustrates the "change of variable" method and essentially represents a leading positive cubic term. (e) is the same "flower" structure shown in (c), but with a "thorn" introduced for each variable. The resulting graph is also bipartite, because the auxiliary variable A can be moved to the same partition as the original variables. The graph now represents the term $-w(1 - X_i)(1 - X_j)(1 - X_k)$, which in effect, is a bipartite representation for an expression with a leading positive cubic term.

graph representations. However, we can construct tripartite graph representations for such terms. In the constructions presented so far, a simple graph-theoretic trick allows us to substitute $(1 - X_l)$ for any of the original Boolean variables X_l. Figure 3(d) shows how this is done for an example variable X_k by introducing an intermediate node with a large weight L associated with it.[3] Assigning unit weights to all original variables and a weight of w to the auxiliary node, a "flower" structure over the variables X_i, X_j and X_k that bears an intermediate node - referred to as the "thorn" - between X_k and the auxiliary node yields a minimum weighted VC of weight $X_i + X_j + X_k + L \cdot (1 - X_k) + w - w \cdot (X_i \cdot X_j \cdot (1 - X_k))$. Treating lower-order terms (recursively) by constructing graphical representations for their negatives, we obtain a graphical structure that essentially represents the positive nonlinear term $+w \cdot (X_i \cdot X_j \cdot X_k)$ as required.[4] In general, the graphical representation for a positive nonlinear term simply falls out of constructing a "flower" structure over the participating variables (using a single auxiliary node), and introducing a "thorn" (intermediate node of large weight) for one of the variables. By the introduction of a "thorn," the graph is no longer bipartite; instead, it becomes tripartite as shown in Figure 3(d).[5]

The only case when the graph is bipartite is when all participating variables in the constraint have a "thorn" associated with them, as shown in Figure 3(e). Here, the auxiliary variable connected to all "thorns" can be given the same "color" as the original variables, that is, they can be in the same partition, hence making the graph bipartite. Such a case yields expressions of the form $-w \cdot (1 - X_i) \cdot (1 - X_j) \cdot (1 - X_k)$ with a positive leading term. We refer to the introduction of "thorns" as the "change of variable" method.

---

[3]Although L is required to be "large," it suffices for it to be greater than the sum of the weights on all other nodes for the graphical representation of that weighted constraint.

[4]Although the above constructions for individual terms allude to treating lower-order terms recursively, any Boolean constraint of arity K can be constructed from a collection of *basis graphs* using at most 2^K auxiliary nodes and 2^K edges (Kumar 2008a).

[5]All original variables should belong to the same partition.

Submodular Constraints

In this section, we will present fast polynomial-time algorithms for solving classes of submodular constraints over Boolean domains. Classes of submodular constraints over Boolean domains that are expressible by graph cuts have been identified in (Zivny and Jeavons 2008). We will show the tractability of the same classes using much simpler arguments, by constructing simple bipartite graphs for the polynomials that represent them. Because our reduction is to flow problems on bipartite graphs, our algorithms have better time complexities.

Corollary 2.10 on page 10 of (Zivny and Jeavons 2008) states that a quadratic polynomial

$$a_0 + \sum_{i=1}^{N} a_i X_i + \sum_{1 \le i < j \le N} a_{ij} X_i X_j$$

over Boolean variables $X_1, X_2 \dots X_N$ represents a submodular cost function if and only if $a_{ij} \le 0$ for every $1 \le i < j \le N$. That is, all nonlinear terms are negative. Using the construction from the previous section, we can obtain bipartite graph representations for all terms. This proves the tractability of the language $\Gamma_{sub,2}$ of all submodular constraints of arity 2 on Boolean domains. The reduction to graph cuts in (Zivny and Jeavons 2008) yields an algorithm that runs in time $O((N + M)^3)$, where N is the number of variables and M is the number of weighted constraints. However, our reduction to minimum weighted VC problems on bipartite graphs yields a more efficient algorithm as max-flow on bipartite graphs is more efficient than on general graphs. In particular, for bipartite graphs, the complexity of max-flow is $O(n_1 m \log(n^2/m))$ where n_1 is the number of nodes in the smaller partition, n is the total number of nodes, and m is the number of edges (Ahuja et al. 1994). In our case, the CCG is a bipartite graph with exactly N nodes in one partition, at most $M + N$ nodes in the other partition, and at most $2M + N$ edges.[6] The time complexity of our algorithm is therefore $O(NM \log M)$. This is better than

---

[6]We assume that the constraint network is connected. If it is not, the problem can be divided into independent subproblems. Thus, we assume that M is greater than $N - 2$.

$O((N + M)^3)$, especially because M could be $O(N^2)$ in the worst case.[7]

Lemma 4.6 on page 14 of (Zivny and Jeavons 2008) states that a cubic polynomial $p(X_1, X_2 \ldots X_N)$ over Boolean variables $X_1, X_2 \ldots X_N$ represents a submodular cost function if and only if it can be written as

$$p(X_1, X_2 \ldots X_N) =$$
$$= a_0 + \sum_{\{i\} \in C_1^+} a_i X_i - \sum_{\{i\} \in C_1^-} a_i X_i$$
$$- \sum_{\{i,j\} \in C_2} a_{ij} X_i X_j$$
$$+ \sum_{\{i,j,k\} \in C_3^+} a_{ijk} X_i X_j X_k - \sum_{\{i,j,k\} \in C_3^-} a_{ijk} X_i X_j X_k$$

where C_2 is the set of quadratic terms, and C_i^+ (C_i^-) is the set of terms of degree $i \in \{1, 3\}$ with non-negative (negative) coefficients and the following two conditions hold:[8]

1. $a_i, a_{ij}, a_{ijk} \geq 0$
   ($\{i\} \in C_1^+ \cup C_1^-$, $\{i,j\} \in C_2$, $\{i,j,k\} \in C_3^+ \cup C_3^-$)

2. $\sum_{k|\{i,j,k\} \in C_3^+} a_{ijk} - a_{ij} \leq 0$
   ($\{i,j\} \in C_2$)

Once again, using the construction from the previous section, we can obtain bipartite graph representations for the linear and negative nonlinear terms. To obtain a bipartite representation for the entire polynomial, the only term that requires further attention is $\sum_{\{i,j,k\} \in C_3^+} a_{ijk} X_i X_j X_k$. We can use the "change of variable" method from the previous section to construct a bipartite graph for

$$\sum_{\{i,j,k\} \in C_3^+} - a_{ijk}(1 - X_i)(1 - X_j)(1 - X_k) =$$
$$\sum_{\{i,j,k\} \in C_3^+} (a_{ijk} X_i X_j X_k - a_{ijk} X_i X_j - a_{ijk} X_j X_k$$
$$- a_{ijk} X_i X_k + a_{ijk} X_i + a_{ijk} X_j + a_{ijk} X_k - a_{ijk})$$

instead, as shown in Figure 3(e).

The leading cubic term is positive as required; the constant term can be ignored; and the linear terms are positive and can be canceled by adding their negatives, which have bipartite graph representations like all linear terms. To cancel the newly introduced quadratic terms, we examine the coefficient that is associated with any quadratic combination $X_i X_j$. This coefficient is $\sum_{k|\{i,j,k\} \in C_3^+} a_{ijk}$ plus the original coefficient $-a_{ij}$ associated with $X_i X_j$. Condition (2) of Lemma 4.6 in (Zivny and Jeavons 2008) requires that $-a_{ij} + \sum_{k|\{i,j,k\} \in C_3^+} a_{ijk} \leq 0$. This means that we have to represent quadratic terms with only negative coefficients,

---

[7]For arity 2, M could be as large as $\binom{N}{2}$.

[8]Condition (2) in (Zivny and Jeavons 2008) has a typo where $-a_{ij}$ has been written as $+a_{ij}$. Note that, except in degenerate cases, having $+a_{ij}$ in the condition does not make sense because this would imply that the sum of non-negative numbers is ≤ 0.

which are already known to have simple "V-structures" as their lifted graphical representations, shown in Figure 3(b).

More formally,

$$p(X_1, X_2 \ldots X_N) =$$
$$= \sum_{\{i,j,k\} \in C_3^+} -a_{ijk}(1 - X_i)(1 - X_j)(1 - X_k)$$
$$+ \sum_{\{i,j,k\} \in C_3^-} -a_{ijk} X_i X_j X_k$$
$$+ \sum_{\{i,j\} \in C_2} [(\sum_{k|\{i,j,k\} \in C_3^+} a_{ijk}) - a_{ij}] X_i X_j$$
$$+ \sum_{\{i\} \in C_1^+} [(\sum_{\{k,j\}|\{i,j,k\} \in C_3^+} -a_{ijk}) + a_i] X_i$$
$$+ \sum_{\{i\} \in C_1^-} [(\sum_{\{k,j\}|\{i,j,k\} \in C_3^+} -a_{ijk}) - a_i] X_i$$
$$+ (\sum_{\{i,j,k\} \in C_3^+} a_{ijk}) + a_0$$

where evidently each term has a bipartite representation. We will refer to this form of the polynomial as its *bipartite representational form*.

There are also necessary - but not sufficient - conditions provided on pages 15-16 of (Zivny and Jeavons 2008) for quartic and higher-order polynomials to be submodular. A simple rearrangement of the terms in these polynomials, similar to the foregoing discussion, shows that they all have bipartite graph representations. For example, Condition (2) in Lemma 5.1 of (Zivny and Jeavons 2008) shows that the positive coefficients of cubic terms and the positive coefficients of quartic terms need to comply in a special way with the coefficients of the quadratic terms. In our framework, this corresponds directly to the facts that: (a) negative nonlinear terms have simple bipartite representations; and (b) only the positive nonlinear terms require special attention, as indicated by C_3^+ and C_4^+ in Condition (2) in Lemma 5.1 of (Zivny and Jeavons 2008). A "change of variable" technique used on the cubic and quartic terms can be used to represent the required positive terms. The extraneous second degree terms that are produced in this process are combined with the original coefficients a_{ij}. The cumulative effect is assured to be non-positive because of Condition (2) in Lemma 5.1 of (Zivny and Jeavons 2008). Therefore, they too have bipartite graph representations. In effect, we can solve the same classes of submodular constraints identified in (Zivny and Jeavons 2008) more efficiently because the underlying max-flow problems are staged on bipartite graphs. For Boolean WCSPs with arity at most K, the bipartite CCG has N nodes in one partition, at most $2^K M$ nodes in the other partition, and at most $K 2^K M$ edges. For K bounded by a constant, this results in a time complexity of $O(NM \log M)$. This significantly improves on the $O((N + M)^3)$ time complexity of the algorithm provided by (Zivny and Jeavons 2008).[9]

---

[9]For arity K, M could be as large as $\binom{N}{K}$.

When Almost All Constraints are Submodular

(Kumar 2008a; 2008b) provide a simple algorithm for constructing tripartite graph representations for an arbitrary weighted constraint of bounded arity. This means that, for a given instance of a Boolean WCSP, we can always construct a CCG for it that is tripartite. Moreover, the complexity of solving this instance is exponential only in the size of the smallest partition - in terms of the number of nodes - of the tripartite CCG constructed for it. This is so because the minimum weighted VC problem can be solved in polynomial time for bipartite graphs; and every possible combination of decisions to include or exclude the nodes of the smallest partition in the VC 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 of the complexity. In particular, when there is sufficient *numerical* structure in the weighted constraints - such as in the submodular constraints discussed above - the CCG is only bipartite, and such classes of WCSPs can be solved in polynomial time. Even when the CCG is not bipartite, our framework allows us to computationally leverage the numerical structure of the weighted constraints.

For example, intuitively, when "most" of the constraints have bipartite representations - like those for the submodular constraints discussed above - the other constraints can still be arbitrary without compromising the tractability. That is, they can be solved in time polynomial in N and M. More precisely, we know that any constraint has a tripartite graph representation; and, if there are only $O(\log N)$ constraints of bounded arity that do not have bipartite representations, then the problem can still be solved in polynomial time. Therefore, the class of all Boolean WCSPs that have submodular constraints of the above kind and a logarithmic number of arbitrary constraints of bounded arity is still tractable.

Planar Constraint Networks

Most of the work done on characterizing the tractability of WCSPs has been on: (1) restricting the nature of the weighted constraints, that is, "language restrictions" and (2) constraining the treewidth of the variable interaction graphs. The CCG provides a unifying framework for exploiting the numerical structure of the weighted constraints as well as the graphical structure of the variable interaction graph since the treewidth of the CCG is identical to that of the variable interaction graph (Kumar 2008a). In this section, we show that the CCG also captures the important structural notion of *planarity* in constraint networks. This again illustrates the usefulness of reformulating combinatorial problems in the form of WCSPs as minimum weighted VC problems on their associated CCGs.

Planar constraint networks are those that can be drawn on a planar surface without any two edges crossing each other. Many combinatorial problems are easier to solve on planar graphs. However, planar graphs do not necessarily have a bounded treewidth.[10] Planarity is thus not captured by a treewidth-based characterization. Very little work has been done on exploiting planarity in constraint networks associated with Boolean WCSPs for computational benefits even though it occurs naturally in real-world domains. Exploiting planarity in conjunction with submodularity is important in various applications too. In computer vision, for example, submodularity arises in semantic segmentation of images, and planarity arises naturally due to the layout (Jegelka and Krause 2012).

Consider the submodular constraints discussed in the previous section (with arity at most 3). If the constraint network is planar, then the CCG is also planar despite the fact that it has additional (auxiliary) variables, as illustrated in Figure 4. An edge between X_i and X_j in the constraint network can arise because of two reasons: (1) There is a binary constraint between X_i and X_j; or (2) X_i, X_j and another variable X_k participate in a ternary constraint. Examining the bipartite representational form of the polynomial $p(X_1, X_2 \ldots X_N)$ which characterizes submodular constraints of arity at most 3, we observe that each binary constraint has a lifted representation with one auxiliary variable as shown in Figure 3(b). In the planar rendition of the lifted representation, this auxiliary node can simply be fitted as an intermediate node on the edge between the two variables in the planar constraint network. This is indicated by the red nodes in Figure 4(b). Each ternary constraint corresponds to a triangle in the constraint network. Once again, examining the bipartite representational form of $p(X_1, X_2 \ldots X_N)$, we observe that the lifted representations for ternary constraints are only of two possible kinds, as shown in Figures 3(c)&(e). In the planar rendition of these lifted representations, the auxiliary nodes can be inscribed within the triangles of the planar constraint network. This is indicated by the blue nodes for positive ternary constraints and the green nodes for negative ternary constraints in Figure 4(b). Thus, the CCG is not only bipartite but also planar.

Now, we can make use of two standard results for planar graphs: (1) The number of edges $|E|$ is at most $3|V| - 6$ for $|V| \geq 3$ vertices; and (2) By Euler's formula for planar graphs, the number of faces $|F|$ is equal to $|E| - |V| + 2$. Since we introduce at most one auxiliary node for every edge and at most 4 auxiliary nodes inscribed in any triangular face of the constraint network, the planar CCG has $O(N)$ nodes and $O(N)$ edges. This means that computing the minimum weighted VC on the bipartite planar CCG requires staging a max-flow on $O(N)$ nodes and $O(N)$ edges. Using the improved max-flow algorithm for such graphs (Orlin 2013), we can solve the problem in time $O(\frac{N^2}{\log N})$. For planar constraint networks, this is a significant improvement over the algorithm presented in (Zivny and Jeavons 2008), which runs in time $O((N + M)^3)$.

Our discussion of exploiting planarity, however, does not have to be in conjunction with submodularity. In fact, the broader $\mathcal{L}_{bipartite}^{Boolean}$ constraints, for arity at most 3, also result in planar CCGs if the constraint networks are planar. This is because the same transformations of edges and trian-

---

[10]An $N \times N$ grid is planar but has treewidth N.

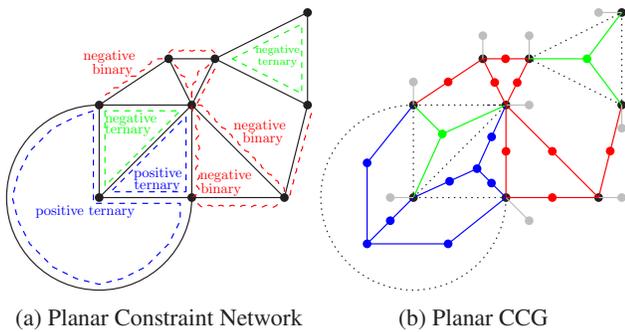(a) Planar Constraint Network   (b) Planar CCG

Figure 4: Shows the invariance of planarity between a constraint network and its CCG. In (a), the red dotted edges indicate negative binary constraints; the blue dotted triangles indicate positive ternary constraints; and the green dotted triangles indicate negative ternary constraints. In (b), the unary constraints are transformed to the grey edges with an auxiliary node; the negative binary constraints are transformed to edges with an intermediate auxiliary node; and the positive and negative ternary constraints are transformed to planar structures inscribed within the corresponding triangles.

gles as described above for submodular constraints continue to apply. Moreover, for general constraints of arity at most 3, planarity of the CCG is still preserved, although it may no longer be bipartite.[11] Nonetheless, finding the minimum weighted VC on a planar graph is amenable to a polynomial-time approximation scheme (PTAS) (Baker 1994).

Conclusions and Future Work

In this paper, we studied submodular constraints because they arise in many real-world applications. We presented fast polynomial-time algorithms for solving classes of submodular constraints over Boolean domains. Reformulating WCSPs as minimum weighted VC problems on their CCGs, we constructed simple bipartite graph representations for the submodular cost functions and translated them into max-flow problems on bipartite graphs. By doing so, we achieved better time complexities than existing algorithms. Furthermore, we also identified tractable classes of WCSPs that have all except a logarithmic number of constraints in $\mathcal{L}_{bipartite}^{Boolean}$. Next, we studied planarity in conjunction with submodularity. Once again, we used the reformulation of WCSPs as minimum weighted VC problems on their CCGs to exploit planarity in order to provide polynomial-time algorithms with significantly improved time complexities. Finally, we discussed planarity outside of submodularity.

In future work, we intend to generalize our techniques to broader classes of WCSPs with higher arities, larger domain sizes, and higher *crossing numbers* of their constraint networks. We would also like to make use of the recent progress on solving max-flow problems, such as exploiting the PTAS

---

[11]Although representing certain terms might require adding newer cancellation terms of lower order, the basis graphs (Kumar 2008a) for representing any constraint of arity at most 3 over Boolean domains are always planar.

for general undirected graphs $G = \langle V, E \rangle$ that runs in time $\tilde{O}((|V| + |E|)^{4/3})$ (Christiano et al. 2011). Moreover, since our algorithms are based on reductions to flow problems, they are amenable to incremental computations that we will further explore.

References

Ahuja, R. K.; B., O. J.; Clifford, S.; and Tarjan, R. E. 1994. Improved algorithms for bipartite network flow. *SIAM Journal on Computing* 23(5):906–933.

Baker, B. S. 1994. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM* 41(1):153–180.

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 Artificial Intelligence Research* 21:135–191.

Christiano, P.; Kelner, J. A.; Madry, A.; Spielman, D. A.; and Teng, S.-H. 2011. Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, 273–282. ACM.

Dechter, R. 2003. *Constraint Processing*. The Morgan Kaufmann Series in Artificial Intelligence. Elsevier Science.

Do, M. B.; Benton, J.; Van Den Briel, M.; and Kambhampati, S. 2007. Planning with goal utility dependencies. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 1872–1878.

Jegelka, S., and Krause, A. 2012. Tutorial on submodularity in machine learning and computer vision. (http://submodularity.org/submodularity-2012.pdf).

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

Krause, A.; Singh, A.; and Guestrin, C. 2008. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research* 9:235–284.

Kumar, T. K. S. 2008a. A framework for hybrid tractability results in boolean weighted constraint satisfaction problems. In *Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming*, 282–297.

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

Orlin, J. B. 2013. Max flows in O(nm) time, or better. In *ACM Symposium on the Theory of Computing*, 765–774.

Sanchez, M.; de Givry, S.; and Schiex, T. 2007. Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. In *Proceedings of the 2007 conference on Artificial Intelligence Research and Development*, 29–37. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Sandholm, T. 2002. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence* 135(1-2):1–54.

Zivny, S., and Jeavons, P. 2008. Classes of submodular constraints expressible by graph cuts. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 112–127.