RC23935 (W0604-059) April 14, 2006
Mathematics

IBM Research Report

Generalized Constraint Satisfaction Problems

Alexander D. Scott
Mathematical Institute
University of Oxford
24-29 St. Giles'
Oxford, OX1 3LB
United Kingdom

Gregory B. Sorkin
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

GENERALIZED CONSTRAINT SATISFACTION PROBLEMS

ALEXANDER D. SCOTT* AND GREGORY B. SORKIN

Abstract. A number of recent authors have given exponential-time algorithms for optimization problems such as Max Cut and Max Independent Set, or for the more general class of Constraint Satisfaction Problems (CSPs). In this paper, we introduce the the class of Generalized Constraint Satisfaction Problems (GCSPs), where the score functions are polynomial-valued rather than real-valued functions. We show that certain reductions used for solving CSPs can be extended to identities for the "partition function" of a GCSP, leading to relatively efficient exponential-time (polynomial-space) algorithms for solving a GCSP. This also enables us (at the cost of only a polynomial factor in time) to modify existing algorithms for optimizing CSPs into algorithms that count solutions or sample uniformly at random. Using an extra variable allows us to solve Max Bisection or calculate the partition function of the Ising Model, problems that were previously inaccessible with this approach.

1. Introduction

A wide range of decision or maximization problems, such as Max Cut, Max k-Cut, and Max 2-SAT belong to the class Max 2-CSP of constraint satisfaction problems with at most two variables per constraint. There is an extensive literature devoted to solving such problems with relatively efficient exponential-time algorithms based on reductions; for example, Gramm, Hirsch, Niedermeier and Rossmanith solve Max 2-SAT in time $\widetilde{O}\left(2^{m/5}\right)$ and use this to solve Max Cut in time $\widetilde{O}\left(2^{m/3}\right)$ [GHNR03], while Kulikov and Fedin solve Max Cut in time $\widetilde{O}\left(2^{m/4}\right)$ [KF02] (where m is the number of constraints or edges).

In this paper, we adopt a novel approach to CSP problems. Rather than working with CSPs as such, we introduce the class of Generalized Constraint Satisfaction Problems (GCSPs), whose scores are given by (products of) polynomials instead of (sums of) real numbers. A GCSP can be thought of as a rather general statistical-mechanical model over a graph, and it is therefore natural to consider the "partition function" of an instance, which sums the (polynomial) scores over all possible assignments.

The advantage of this approach comes when we consider the relationship between CSPs and GCSPs. Given a CSP, there is a natural way to obtain a corresponding GCSP. The partition function of the GCSP then corresponds to the generating function for (scores of) assignments in the original CSP (and, conversely, the CSP corresponds to the "Hamiltonian" of the GCSP). Thus calculating the partition function of the GCSP not only gives us the optimal score for the CSP, but tells us how many assignments achieve each possible score. This enables us to count solutions, and can be used to sample at random from solutions. (The beautifully simple new algorithm of Ryan Williams [Wil04] can also be used to count and sample; taking an approach completely different from ours, it only delivers an approximation for real-valued scores, and requires exponential space as well as exponential time.)

A standard approach to dealing with 2-CSP problems (which we will henceforth just call CSPs), especially in the sparse case, is to define a sequence of reductions each of which replaces a given problem instance by one or more "smaller" instances that are in some sense equivalent to the original instance. The original instance can then be solved by working recursively on the reduced instances. Surprisingly, it turns out that a number of existing CSP reductions can be carried through into the framework of GCSPs, where they can be extended to an identity between partition functions of two or more GCSPs (the original reductions correspond to indentities between leading terms of partition functions).

* Research supported in part by EPSRC grant GR/S26323/01.

We can therefore translate some existing reduction-based algorithms for CSP (notably those from [SS03, SS06, SS04]) to similarly efficient algorithms for calculating the partition function, thus delivering a substantial amount of additional information. For instance, if the score function is integral and polynomially bounded, we can count the number of solutions with each possible total score; for arbitrary score functions, we can count the number of optimal solutions; we can solve optimization problems with multiple constraints and/or objectives, such as Max Bisection; and in any of these cases we can sample solutions at random.

2. Outline

Section 3 defines the classes CSP and Generalized CSP. Section 4 provides examples of CSPs and related GCSPs; because this includes problems that could not previously be solved by methods of this kind, such as Max Bisection, it is a focal point of the paper. In Section 5, we define our reductions. These are extensions into the GCSP framework of reductions that have previously been used for CSPs; in this context they give identities between partition functions. In Section 6 we show algorithmic consequences for CSP instances with polynomially bounded integer scores, and in Section 7 we extend this to allow arbitrary real scores. In Section 8, we state a variety of results for determining a CSP's optimal score, and counting solutions with this and other scores; in Section 9 we note that we can also produce (and randomly sample) the solutions themselves.

Notation: In the next section we will define the class of (simple) CSPs and our new class GCSP of Generalized CSPs. An instance of either will utilize an underlying graph $G = (V, E)$ (with vertex set V and edge set E), and we will reserve the symbols G, V and E for these roles. An instance of either also has a domain of *values* or *colors* that may be assigned to the vertices (variables), for example $\{\text{true}, \text{false}\}$ for a satisfiability problem, or a set of colors for a graph coloring problem. In general we will denote this domain by $[k]$, interpreted as $\{0, \dots, k-1\}$ or (it makes no difference) $\{1, \dots, k\}$. At the heart of both of the CSP and GCSP instance will be cost or "score" terms s_v^i ($v \in V(G)$, $i \in [k]$) and s_{xy}^{ij} ($x, y \in V(G)$, $i, j \in [k]$). *These superscripts denote indices, not exponentiation!*

For CSPs the scores s are real numbers, while for GCSPs they are polynomials (we will typically write these in the variable z if univariate, or if multivariate over z, w or $z, w_1, w_2, \dots$). While the notational confusion is unfortunate, a *superscripted* s always indicates an *index*, while a *superscripted z or w* always indicates a *power*. In cases where the CSP scores s are not positive integers, the exponents in the GCSP score "polynomials" will likewise be fractional or negative (or both).

We use $\uplus$ to indicate disjoint union, so $V_0 \uplus V_1 = V$ means that V_0 and V_1 partition V, i.e., $V_0 \cap V_1 = \emptyset$ and $V_0 \cup V_1 = V$. The notation $\widetilde{O}$ hides polynomial factors in any parameters, so for example $O(n2^{19m/100})$ is $\widetilde{O}\left(2^{19m/100}\right)$.

3. CSP and Generalized CSP

Let us begin by defining the problem class CSP over a domain of size k. An *instance I of CSP with underlying graph $G = (V, E)$ and domain* $[k]$ has the following ingredients:

(1) a real number $s_\emptyset$;
(2) for each vertex $v \in V$ and color $i \in [k]$, a real number s_v^i;
(3) for each edge $xy \in E$ and any colors $i, j \in [k]$, a real number s_{xy}^{ij}.

We shall refer to these quantities as, respectively, the *nullary score*, the *vertex scores*, and the *edge scores*. Note that we want only one score for a given edge with given colors assigned its endpoints, so s_{xy}^{ij} and s_{yx}^{ji} are taken to be equivalent names for the same score (or one may simply assume that $x < y$).

Given an *assignment* (or *coloring*) $\sigma : V \to [k]$, we define the *score of* σ to be the real number

$$I(\sigma) := s_\emptyset + \sum_{v \in V} s_v^{\sigma(v)} + \sum_{xy \in E} s_{xy}^{\sigma(x)\sigma(y)}.$$

We want to count the number of solutions satisfying various properties, and therefore take a generating function approach. Given an instance I, the corresponding generating function is the polynomial

$$\sum_{\sigma:V\to[k]} z^{I(\sigma)}. \tag{1}$$

More generally, if we want to keep track of several quantities simultaneously, say $I(\sigma), J(\sigma), \cdots$, we can consider a multivariate generating function $\sum_\sigma z^{I(\sigma)} w^{J(\sigma)} \cdots$. Calculating the generating function in the obvious way (by running through all $k^{|G|}$ assignments) is clearly very slow, so it is desirable to have more efficient algorithms.

In order to handle generating functions, we borrow some notions from statistical physics. We think of the score $I(\sigma)$ as a "Hamiltonian" measuring the "energy" of a configuration σ. Thus the edge scores correspond to "pair interactions" between adjacent sites, while the vertex scores measure the effect of adding a "magnetic field". (From this perspective, the nullary score is just a constant that disappears after normalization.) The generating function is then the "partition function" for this model.[1]

A crucial element of our approach is that the score $I(\sigma)$ can be broken up as a sum of local interactions, and thus an expression such as $z^{I(\sigma)}$ can be expressed as a product of monomials corresponding to local interactions. In order to provide a framework for this approach, we introduce a generalized version of constraint satisfaction where the scores are polynomials (in some set of variables) instead of real numbers, and the score of an assignment is taken as a product rather than a sum.

An *instance I of Generalized CSP, with underlying graph G and domain* $[k]$, has the following ingredients:

(1) a polynomial $p_\emptyset$;
(2) for each vertex $v \in V$ and color $i \in [k]$, a polynomial p_v^i;
(3) for each edge $xy \in E$ and pair of colors $i, j \in [k]$, a polynomial p_{xy}^{ij}.

We shall refer to these three types of polynomial as, respectively, the *nullary polynomial*, the *vertex polynomials*, and the *edge polynomials*. We want only one polynomial for a given edge with given colors on its endpoints, so again we either take p_{xy}^{ij} and p_{yx}^{ji} to be equivalent or simply assume that $x < y$.

Given an assignment $\sigma : V \to [k]$, we define the *score of* σ to be the polynomial

$$I(\sigma) := p_\emptyset \cdot \prod_{v\in V} p_v^{\sigma(v)} \cdot \prod_{xy\in E} p_{xy}^{\sigma(x)\sigma(y)}.$$

We then define the *partition function Z_I of I* by

$$Z_I = \sum_{\sigma:\, V\to[k]} I(\sigma).$$

This should become clearer in the next section when we give a few examples. In particular, we note that for every CSP there is a naturally corresponding GCSP, and we indicate how different problems can be translated into the GCSP context.

We will turn to algorithms in subsequent sections, but it is worth bearing in mind that Generalized CSP over a graph G can be solved about as efficiently as a CSP over the same graph: as we will see, we pay only a polynomial multiplicative factor for the "bookkeeping" of working with polynomials rather than numbers.

4. Examples

In this section we show how some standard problems can be written in terms of instances of Generalized CSP. For convenience we sometimes call an instance I of GCSP a *generalized instance*.

[1] The partition function is usually written in the form $\sum_\sigma \exp(-\beta I(\sigma))$, where β is known as the *inverse temperature*, but substituting z for $e^{-\beta}$ yields the partition function in polynomial form.

4.1. Generating function of a simple CSP.

Definition 1 (Generating function of an instance). *Given an instance I of CSP, we can define a corresponding instance I^* of GCSP with the same underlying graph and variable domain, and polynomials*

$$\begin{aligned} p_\emptyset &= z^{s_\emptyset} & \\ p_v^i &= z^{s_v^i} & i \in [k] \\ p_{xy}^{ij} &= z^{s_{xy}^{ij}} & i,j \in [k] \end{aligned}$$

The connection between I and I^* is given by the following simple observation.

Lemma 2. *Let I be an instance of CSP, and let I^* be the corresponding generalized instance. Then the partition function Z_{I^*} is the generating function (1) for the instance I.*

Proof. For any assignment σ, we have

$$I^*(\sigma) = z^{s_\emptyset} \cdot \prod_{v\in V} z^{s_v^i} \cdot \prod_{xy\in E} z^{s_{xy}^{ij}} = z^{s_\emptyset + \sum_{v\in V} s_v^i + \sum_{xy\in E} s_{xy}^{ij}} = z^{I(\sigma)}.$$

It therefore follows that the partition function $Z_{I^*} = \sum_\sigma z^{I(\sigma)}$ is the generating function for the original constraint satisfaction problem. □

Similar results are easily seen to hold for generating functions in more than one variable.

4.2. Max Cut and Max Dicut.

Max Cut provides a simple illustration of Definition 1 and Lemma 2. Let us first write Max Cut as a CSP; we will then construct the corresponding GCSP.

Example 3 (Max Cut CSP). *Given a graph $G = (V,E)$, set $k = 2$ and define a CSP instance I by*

$$s_\emptyset = 0 \quad ; \quad (\forall v \in V) \quad s_v^0 = s_v^1 = 0 \quad ; \quad (\forall xy \in E) \quad s_{xy}^{10} = s_{xy}^{01} = 1,\ s_{xy}^{00} = s_{xy}^{11} = 0.$$

With $\sigma^{-1}(i) = \{v \colon \sigma(v) = i\}$, note that $(V_0, V_1) = (\sigma^{-1}(0), \sigma^{-1}(1))$ is a partition of V, and

$$I(\sigma) = \sum_{\substack{xy\in E:\\ \sigma(x)=0,\, \sigma(y)=1}} 1 = e(V_0, V_1) \tag{2}$$

is the size of the cut induced by σ. The corresponding GCSP instance is obtained as in Definition 1.

Example 4 (Max Cut GCSP). *Given a graph $G = (V,E)$, we set $k = 2$ and define a GCSP instance I by*

$$s_\emptyset = 1 \quad ; \quad (\forall v \in V) \quad s_v^0 = s_v^1 = 1 \quad ; \quad (\forall xy \in E) \quad s_{xy}^{10} = s_{xy}^{01} = z,\ s_{xy}^{00} = s_{xy}^{11} = 1.$$

(In all such cases, a 1 on the right hand side may be thought of as z^0.)

By Lemma 2, the partition function Z_I is therefore the generating function for cuts:

$$Z_I = \sum 2c_i z^i, \tag{3}$$

where c_i is the number of cuts of size i, and the factor 2 appears because each cut (V_0, V_1) also appears as (V_1, V_0). The size of a maximum cut is the degree of Z_I, and the number of maximum cuts is half the leading coefficient.

Note that the partition function (3) is the partition function of the Ising model with no external field (see below for a definition). Thus we have recovered the very familiar fact that, up to a change of variables, the partition function of the Ising model is the generating function for cuts.

We can also encode weighted instances of Max Cut with edge weights $w : E \to \mathbb{R}$ by modifying the third line of the definition above to

$$p_{xy}^{10} = p_{xy}^{01} = z^{w(xy)} \qquad \forall xy \in E.$$

The Max Dicut CSP and GCSPs are encoded essentially identically, as is Max k-Cut.

4.3. **The Ising model and Max Bisection.** A slight generalization of the previous example allows us to handle the Ising model.

Example 5 (Ising model). *The Ising model with edge weights J and external field h on a graph G is defined in terms of its Hamiltonian H. For an assignment $\sigma : V \to \{0, 1\}$, we define*

$$H(\sigma) = J \sum_{xy \in E} \delta(\sigma(x), \sigma(y)) + h \sum_{v \in V} \sigma(v).$$

Here $\delta(a, b)$ is the delta function, returning 0 if $a = b$, and 1 otherwise, J is the *interaction strength* and h is the *external magnetic field.* In analogy with (2), taking $V_i = \sigma^{-1}(i)$, we may rewrite H as

$$H(\sigma) = Je(V_0, V_1) + h|V_1|.$$

Note that H is an instance of (non-generalized) CSP.

The *partition function* of the Ising model at *inverse temperature* β is

$$Z_{\text{Ising}} = \sum_{\sigma} e^{-\beta H(\sigma)} = \sum_{V_0 \uplus V_1 = V} w^{|V_1|} z^{e(V_0, V_1)}, \tag{4}$$

where we have written $w = e^{-\beta h}$ and $z = e^{-\beta J}$, and the last sum is taken over ordered pairs (V_0, V_1) that partition V. With this change of variables, the Ising partition function is easily expressed as partition function of a GCSP, this time over two variables.

Example 6 (Ising GCSP). *Define an Ising GCSP instance I by*

$$p_\emptyset = 1 \quad ; \quad (\forall v \in V) \quad p_v^0 = 1,\ p_v^1 = w \quad ; \quad (\forall xy \in E) \quad p_{xy}^{10} = p_{xy}^{01} = z,\ p_{xy}^{00} = p_{xy}^{11} = 1.$$

With our usual notation, $I(\sigma) = w^{|V_1|} z^{e(V_0, V_1)}$, and so the partition function for this GCSP is equal to (4).

The Ising GCSP can also be used to handle *Max Bisection.* This is important, because CSP algorithms such as those in [GHNR03, KF02, SS03, SS04] solve Max Cut, but cannot be applied to Max Bisection because there is no way to force them to generate a balanced cut. At the modest expense of a polynomial factor in the running time, the Ising GCSP does this by tracking two variables at once: the bisections of G correspond to terms in Z_{Ising} that have degree $\lfloor n/2 \rfloor$ in w. (Each bisection is counted once if n is odd, twice if n is even.) Extracting these terms gives the generating function for bisections of G.

The same partition function also yields a *sparsest cut* of the graph: sparsest cuts correspond to terms with the largest ratio of the power of z (number of cut edges) to the power of w (number of vertices in one partition).

4.4. **Max Independent Set and Max Clique.** Maximum Independent Set (MIS) is easily expressed as a 2-CSP:

$$s_\emptyset = 0 \quad ; \quad (\forall v \in V) \quad s_v^0 = 0,\ s_v^1 = 1 \quad ; \quad (\forall xy \in E) \quad s_{xy}^{00} = s_{xy}^{10} = s_{xy}^{01} = 0,\ s_{xy}^{11} = -2.$$

Maximum clique cannot be modeled in the same way, because the clique constraint is enforced by anti-edges, which are not an element of the model. Of course a maximum clique in G corresponds to a maximum independent set in its complement graph $\bar{G}$, but for our purposes this is very different, as the running time of our algorithm is parametrized by the number of edges, and a sparse Max Clique instance becomes a dense MIS instance. However, as with Max Bisection, it is possible to model a sparse Max Clique instance as a GCSP with the same input length by introducing a second variable:

$$s_\emptyset = 1 \quad ; \quad (\forall v \in V) \quad s_v^0 = 1,\ s_v^1 = w \quad ; \quad (\forall xy \in E) \quad s_{xy}^{00} = s_{xy}^{10} = s_{xy}^{01} = 1,\ s_{xy}^{11} = z$$

A coloring σ has score $I(\sigma) = w^{|V_1|} z^{e(G|_{V_1})}$, the power of w counting the number of vertices in the chosen set $V_1 = \sigma^{-1}(1)$, and the power of z counting the number of edges induced by that set. In the partition function, k-cliques correspond to terms of the form $w^k z^{\binom{k}{2}}$. Of course, independent sets of cardinality k correspond to terms with $w^k z^0$: the GCSP simultaneously counts maximum cliques and maximum independent sets (among other things).

4.5. **Judicious partitions and simultaneous assignments.** As is spelled out in the full version of the paper, similar techniques may be applied to various *judicious partitioning* problems [BS99, Sco05], such as finding a cut of a graph which minimizes $\max\{e(V_0), e(V_1)\}$. We think of this as a case of having more than one CSP on a single set of variables. In other examples, we might have two instances of Max SAT that we wish to treat as a bi-criterion optimization problem; a Max SAT instance to optimize subject to satisfaction of a SAT instance; or a partition of a vertex set yielding a large cut for two different graphs on the same vertices. All such problems are now straightforward: use a variable z to encode one problem and a variable w to encode the second. Initial edge scores would thus be of the form $z^{\text{firstscore}} w^{\text{secondscore}}$.

5. Reductions

In this section we introduce several types of reduction. A Type 0 reduction expresses the partition function of a GCSP as a product of partition functions of two smaller instances (or in an important special case, a single such instance). Type I and Type II reductions each equate the generating function of an instance to that of an instance with one vertex less. Finally, a Type III reduction produces k instances, whose partition functions sum to the partition function of the original instance. In each case, once the reduction is written down, verifying its validity is a simple matter of checking a straightforward identity.

The four reductions correspond to the CSP reductions used in [SS03, SS06, SS04]. However, by working with polynomials rather than real numbers, we are able to carry substantially more information. A word of intuition, deriving from the earlier CSP reductions, may be helpful (though some readers may prefer to go straight to the equations). A CSP II-reduction was performed on a vertex v of degree 2, with neighbors u and w. The key observation is that in a maximization problem, the optimal assignment $\sigma(v)$ is a function of the assignments $\sigma(u)$ and $\sigma(v)$. That is, given any assignments $\sigma(u) = i$ and $\sigma(w) = j$, the optimal combined scores of the vertex v, the edges uv and vw, and the edge uw (if present, and otherwise taken to be the 0 score function) is

$$\tilde{s}^{ij}_{uw} = \max_l \left\{ s^{ij}_{uw} + s^{il}_{uv} + s_v(l) + s^{lj}_{vw} + \right\}. \tag{5}$$

By deleting v from the CSP instance and replacing the original score s_{uw} with the score $\tilde{s}_{uw}$, we obtain a smaller instance with the same maximum value. In the GCSP context we do essentially the same thing, except that the maximization in (5) is replaced by the summation in (7) (and the sum replaced by a product).

Type 0 Reduction. Suppose I is a GCSP instance whose underlying graph G is disconnected. Let $V = V_0 \cup V_1$ be a nontrivial partition such that $e(V_1, V_2) = 0$. Let I_1 and I_2 be the subinstances obtained by restriction to V_1 and V_2, except that we define the nullary polynomials by

$$p^{(I_1)}_\emptyset = p^{(I)}_\emptyset \qquad ; \qquad p^{(I_2)}_\emptyset = 1.$$

A straightforward calculation shows that, for any assignment $\sigma : V \to [k]$, we have

$$\begin{aligned} I(\sigma) &= p_\emptyset \cdot \prod_{v \in V} p_v^{\sigma(v)} \cdot \prod_{xy \in E} p_{xy}^{\sigma(x)\sigma(y)} \\ &= \left(p_\emptyset \cdot \prod_{v \in V_1} p_v^{\sigma(v)} \cdot \prod_{xy \in E(G[V_1])} p_{xy}^{\sigma(x)\sigma(y)} \right) \left(1 \cdot \prod_{v \in V_2} p_v^{\sigma(v)} \cdot \prod_{xy \in E(G[V_2])} p_{xy}^{\sigma(x)\sigma(y)} \right) \\ &= I_1(\sigma_1) I_2(\sigma_2), \end{aligned}$$

where σ_i denotes the restriction of σ to V_i. It follows easily that $Z_I = Z_{I_1} Z_{I_2}$. Thus in order to calculate Z_I it suffices to calculate Z_{I_1} and Z_{I_2}.

When one component of G is an isolated vertex v, with $V' = V \setminus v$, one term becomes trivial: $Z_I = \left(p_\emptyset \cdot \sum_{i\in[k]} p_v^i\right) \cdot Z_{I_2}$. So $Z_I = Z_{\widetilde{I}}$, where $\widetilde{I}$ is the instance obtained from I by deleting v, defining $p_\emptyset^{\widetilde{I}} = p_\emptyset \cdot \sum_{i\in[k]} p_v^i$ (a trivial calculation), and leaving all other scores unchanged.

Type I Reduction. Suppose that I is an instance with underlying graph G, and $v \in V$ has degree 1. Let w be the neighbor of v. We shall replace I by an "equivalent" instance $\widetilde{I}$ (one with the same partition function) with underlying graph $G \setminus v$.

We define the instance $\widetilde{I}$ by giving w vertex scores

$$\widetilde{p}_w^i = p_w^i \sum_{j=0}^{k-1} (p_{wv}^{ij} \cdot p_v^j). \tag{6}$$

All other scores remain unchanged (except for p_v^* and p_{vw}^{**}, which are deleted along with v).

To show that $Z_I = Z_{\widetilde{I}}$, let $\sigma : V \backslash v \to [k]$ be any assignment and, for $j \in [k]$, extend σ to $\sigma^j : V \to [k]$ defined by $\sigma^j(v) = j$ and $\sigma^j|_{V\backslash v} = \sigma$. Using (6), we have

$$\begin{aligned}
\widetilde{I}(\sigma) &= \widetilde{p}_\emptyset \cdot \prod_{x\in V\backslash v} \widetilde{p}_x^{\sigma(x)} \cdot \prod_{xy\in E(G\backslash v)} \widetilde{p}_{xy}^{\sigma(x)\sigma(y)} \\
&= \left(p_\emptyset \cdot \prod_{x\in V\backslash\{v,w\}} p_x^{\sigma(x)} \cdot \prod_{xy\in E(G\backslash v)} p_{xy}^{\sigma(x)\sigma(y)} \right) \cdot \left(p_w^{\sigma(w)} \cdot \sum_{j=0}^{k-1} p_{wv}^{\sigma(w)j} \cdot p_v^j \right) \\
&= \sum_{j=0}^{k-1} p_\emptyset \cdot \prod_{v\in V} p_v^{\sigma^j(v)} \cdot \prod_{xy\in E} p_{xy}^{\sigma^j(x)\sigma^j(y)} \\
&= \sum_{j=0}^{k-1} I(\sigma^j)
\end{aligned}$$

and so $Z_I = \sum_{\sigma:V\to[k]} I(\sigma) = \sum_{\sigma:V\backslash v\to[k]} \sum_{j=0}^{k-1} I(\sigma^j) = \sum_{\sigma:V\backslash v\to[k]} \widetilde{I}(\sigma) = Z_{\widetilde{I}}$.

Type II Reduction. Suppose that I is an instance with underlying graph G, and $v \in V$ has degree 2. Let u and w be the neighbors of v in G. We define an instance $\widetilde{I}$ with underlying graph $\widetilde{G}$, which will have fewer vertices and edges than G. The underlying graph $\widetilde{G}$ of $\widetilde{I}$ is obtained from G by deleting v and adding an edge uw (if the edge is not already present). We then define $\widetilde{I}$ by setting, for $0 \le i, j \le k-1$,

$$\widetilde{p}_{uw}^{ij} = p_{uw}^{ij} \cdot \sum_{l=0}^{k-1} p_{uv}^{il} p_v^l p_{vw}^{lj}, \tag{7}$$

where we take $p_{uw}^{ij} = 1$ if it is not already defined. All other scores remain unchanged (except for p_v^*, p_{vu}^{**} and p_{vw}^{**}, which are deleted).

To show that $Z_I = Z_{\widetilde{I}}$, let $\sigma : V \setminus v \to [k]$ be any assignment. As before, we write σ^l for the assignment with $\sigma^l|_{V\backslash v} \equiv \sigma$ and $\sigma^l(v) = l$. Then

$$\begin{aligned}
\widetilde{I}(\sigma) &= \widetilde{p}_\emptyset \cdot \prod_{x\in V\backslash v} \widetilde{p}_x^{\sigma(x)} \cdot \prod_{xy\in E(\widetilde{G})} \widetilde{p}_{xy}^{\sigma(x)\sigma(y)} \\
&= p_\emptyset \prod_{x\in V\backslash v} p_x^{\sigma(x)} \cdot \prod_{xy\in E(G\backslash v)} p_{xy}^{\sigma(i)\sigma(j)} \cdot \sum_{l=0}^{k-1} p_{uv}^{\sigma(u)l} p_v^l p_{vw}^{l\sigma(w)} \\
&= \sum_{l=0}^{k-1} I(\sigma^l).
\end{aligned}$$

As with Type I reductions, this implies that $Z_I = Z_{\widetilde{I}}$.

Type III Reduction. Suppose that I is an instance with underlying graph G, and $v \in V$ has degree 3 or more. Let $u_1, \dots, u_d$ be the neighbors of v in G. We define k instances $\widetilde{I}_0, \dots, \widetilde{I}_{k-1}$ each with underlying graph $\widetilde{G} = G \setminus v$. For $i = 0, \dots, k-1$, the ith instance $\widetilde{I}_i$ corresponds to the set of assignments where we have taken $\sigma(v) = i$.

We define nullary scores for $\widetilde{I}_i$ by setting

$$q_\emptyset^{(i)} = p_\emptyset \cdot p_v^i$$

and, for $1 \le l \le d$ and $0 \le j \le k-1$, vertex scores

$$(q^{(i)})_{u_l}^j = p_{u_l}^j \cdot p_{u_l v}^{ji}.$$

All other scores remain unchanged from I (except for p_v^* and p_{v*}^{**}, which are deleted).

For any assignment $\sigma : V \setminus v \to [k]$, and $0 \le i \le k-1$, we write (as usual) $\sigma^i : V \to [k]$ for the assignment with $\sigma^i|_{V \setminus v} \equiv \sigma$ and $\sigma^i(v) = i$. Then, writing $W = V \setminus (v \cup \Gamma(v))$,

$$\begin{aligned} I(\sigma^i) &= p_\emptyset \prod_{x \in V} p_x^{\sigma^i(x)} \cdot \prod_{xy \in E} p_{xy}^{\sigma^i(x)\sigma^i(y)} \\ &= p_\emptyset \prod_{x \in W} p_x^{\sigma(x)} \cdot \prod_{xy \in E(G \setminus v)} p_{xy}^{\sigma(x)\sigma(y)} \cdot p_v^i \prod_{l=1}^{d} p_{u_l}^{\sigma(u_l)} p_{u_l v}^{\sigma(u_l) i} \\ &= \widetilde{I}_i(\sigma), \end{aligned}$$

since $p_v^{\sigma^i(v)} = p_v^i$. Then $Z_I = \sum_{\sigma : V \to [k]} I(\sigma) = \sum_{\sigma : V \setminus v \to [k]} \sum_{i=0}^{k-1} I(\sigma^i) = \sum_{\sigma : V \setminus v \to [k]} \sum_{i=0}^{k-1} \widetilde{I}_i(\sigma) = \sum_{i=0}^{k-1} \sum_{\sigma : V \setminus v \to [k]} \widetilde{I}_i(\sigma) = \sum_{i=0}^{k-1} Z_{\widetilde{I}_i}$. Thus the partition function for I is the sum of the partition functions for the $\widetilde{I}_i$.

6. Working with polynomially bounded instances

In this section we consider algorithms for calculating the partition function of an instance of GCSP. The basic idea is that we begin with an instance I and apply a sequence of reductions to reduce it to successively smaller instances. If we can apply a reduction of Type 0 (to an isolated vertex), Type I or Type II then the instance is replaced with an equivalent instance I' with fewer vertices than I. Otherwise, we perform a Type III reduction, and replace I with k smaller instances $I_0, \dots, I_{k-1}$, such that the partition function of I equals the sum of the partition functions of the k smaller instances. We recursively solve the instances $I_0, \dots, I_{k-1}$ and sum the partition functions to obtain I. This approach was used in the algorithms for MAX CSP analyzed in [SS06, SS03, SS04]. Our aim in this section is to extend the results from these papers into the generalized context.

We begin by giving bounds on the time required to effect reductions of different types. We deal first with the single-variable case. We use the fact that two polynomials of degree d can be multiplied in time $O(d \log d)$.

Lemma 7. *Let I be an instance of GCSP with underlying graph G (with n vertices and m edges) and domain size k. Suppose that the scores are polynomials in the variable z, and the maximum degree of any score polynomial is c. Let I^* be any instance obtained from I by a sequence of 0, I and II reductions, and let $D = c(m + n + 1)$. Then a Type 0 reduction on a single vertex can be performed in time $O(kD + D \log D)$, a Type I reduction in time $O(k^2 D \log D)$, a Type II reduction in time $O(k^3 D \log D)$, and a Type III reduction on a vertex of degree d in time $O(k^3 dD \log D)$.*

Proof. Note first that any score function for an edge or vertex of I^* is a sum of products of score polynomials from I. Furthermore, each such product includes at most one score polynomial from each edge or vertex of I. Thus the degree of any score function in I^* is at most D (corresponding to the edge, vertex and nullary scores from I^*).

Performing a Type 0 reduction on a single vertex requires us to sum k score polynomials and multiply by a $(k+1)$st. This can be done in time $O(kD) + O(D \log D)$. We reason similarly for Type I, II and III reductions. □

A similar result holds in the multivariable case. We can now carry across theorems from previous papers. The reductions for generalized instances work in the same way as for simple instances of Max CSP, except with an additional time factor. For instance, the expected linear time result Theorem 7.1 of [SS06] becomes the following.

Theorem 8. *For any $\lambda = \lambda(n)$ and any $\eta \leq 1 + \lambda n^{-1/3}$, let $G \in \mathcal{G}(n, c/n)$ be a random graph and let I be any GCSP instance over G, with domain size 2. Let c be the maximum degree of any score polynomial in I. Then the partition function of I can be calculated in expected time $O(n)\exp(1+\lambda^3) \cdot cn^2 \log(cn)$.*

From Theorem 5 of [SS04], we get:

Theorem 9. *Let G be a graph with n vertices and m edges, and let I be any GCSP instance over G in a variable z, where the domain size is k and the maximum degree of any score function is c. Then the partition function of I can be calculated in time $O(k^3 c(m+n)\log(cn)) \cdot k^{19m/100}$.*

7. A reduced algorithm for instances with arbitrary weights

In this section we drop the assumption that our GCSP score function powers (corresponding to CSP weights) are polynomially bounded integers, and instead work with arbitrary weights. The problem now is that our "polynomials" might have exponentially many terms, and we are therefore compelled to "prune" them. ("Polynomial" is now a loose term, as we may allow non-integral and negative powers.)

Pruning technique. Given a polynomial p in one variable z, we define the *pruned polynomial* p_z to be the polynomial obtained by removing all but the leading term. If p is a polynomial in variables $z, w_1, w_2, \ldots$, we obtain p_z by throwing away all terms t such that there is a term of the form ctz^i for some $i > 0$. For instance,

$$(2z^2 + 3z - 700 + zw_1 + z^2 w_1 + zw_2 + z^{10} w_1 w_2)_z = 2z^2 + z^2 w_1 + zw_2 + z^{10} w_1 w_2.$$

Given an instance I, the *z-pruned* (or simply "pruned") *instance* I_z is obtained from I by replacing all score functions by their pruned equivalents. The point about pruning is that, for maximization problems in z alone, I and I_z have the same value. More generally, if there are variables $w_1, w_2, \ldots$ as well as z, then pruning removes only terms that do not contribute to the maximum value indexed by z, for any given values of the parameters indexed by the other variables.

That is, z-pruning the final generating function Z_I leaves a generating function $(Z_I)_z$ for solutions maximizing the power of z. The algorithmically essential point is that we can prune as we reduce. This is expressed by the following lemma, but the key point is simply that for polynomials p and q (in any set of variables) $(pq)_z = (p_z q_z)_z$.

Lemma 10. *For any reduction R from our list, and any instance I, $R(I)_z = R(I_z)_z$. In particular, if we perform a full sequence of reductions and z-prune at every stage, we will end up with $(Z_I)_z$.*

Thus for maximization problems, we can carry across our theorems to instances with arbitrary weights by pruning our model at every stage. Since this takes only polynomial time we retain our time bounds.

Theorem 11. *Let G be a graph with n vertices and m edges, and let I be any GCSP instance over G in variables $z, w_1, \ldots, w_r$, where the domain size is k, the maximum degree of any score function is c, and we allow nonintegral powers of z in the score functions. Then the z-pruned partition function of I can be calculated in time $\widetilde{O}((m+n)^r c^r k^{19m/100})$.*

8. Applications

We have shown in [SS04] that Max Cut can be solved in time $\widetilde{O}\left(2^{19m/100}\right)$.

Corollary 12. *Let G be a graph with m edges. The partition function of the Ising model on G with edge interactions w and external magnetic field h can be calculated in time $\widetilde{O}(2^{19m/100})$. In particular, Max Bisection and Min Bisection can be solved in the same time.*

Corollary 13. *Let G be a graph with m edges. In time $\widetilde{O}(2^{19m/100})$, we can solve Max Clique and Max Independent Set, and count cliques and independent sets of all sizes.*

For weighted versions of the problems above, we can use the algorithm from Section 7. For example where [SS04] solved weighted Max Cut in a graph with arbitrary real edge weights, here we can count the maximizing solutions.

Corollary 14. *Let G be a edge-weighted graph with m edges of arbitrary real weights. Then in time $\widetilde{O}(2^{19m/100})$ we can solve Max Cut, Max Bisection and Min Bisection, and count the optimal cuts and bisections.*

Corollary 15. *Let G be a vertex-weighted graph, with arbitrary real weights, having m edges. For $K > 0$, in time $\widetilde{O}(2^{19m/100})$, we can find the maximum weight of a clique of order K or an independent set of size K and count such cliques or independent sets. In particular, we can solve Max Weighted Independent Set and count its solutions.*

9. Constructing and sampling solutions

Finally, wherever we can compute a CSP's maximum value, we can produce a corresponding assignment; and wherever we can count assignments producing a given value, we can also do exact random sampling from these assignments. The method is standard, and we illustrate with sampling. We construct our assignment one variable at a time, starting from the empty assignment. Given a partial assignment $\sigma_0 : V_0 \to [k]$, and a vertex $v \notin I_0$, we calculate the partition functions

$$Z_{I;\sigma_0^i} = \sum_{\sigma\colon \sigma|_{V_0}=\sigma_0,\ \sigma(v)=i} I(\sigma),$$

and use these to determine the conditional distribution of $\sigma(v)$ given that $\sigma|_{I_0} = \sigma_0$.

This enables us to sample from a variety of distributions. For instance, we get the following result.

Theorem 16. *Let G be a graph with m edges. Then in time $\widetilde{O}(2^{19m/100})$ we can sample uniformly at random from the following distributions:*

- *maximum cuts, maximum bisections, minimum bisections*
- *maximum independent sets, cliques of maximal size*
- *independent sets of any fixed size, cliques of any fixed size*

In the same time we can sample from the equilibrium (Gibbs) distribution of the Ising model with any fixed interaction strength and external magnetic field.

A similar result holds for edge- or vertex-weighted graphs, except that we only sample at random from optimal assignments.

Many other problems can be expressed in this framework. For example, we can count and sample proper k-colorings in time $\widetilde{O}\left(k^{19m/100}\right)$.

References

[BS99] B. Bollobás and A. D. Scott, *Exact bounds for judicious partitions of graphs*, Combinatorica **19** (1999), no. 4, 473–486. MR **MR1773653 (2001f:**05083)

[GHNR03] Jens Gramm, Edward A. Hirsch, Rolf Niedermeier, and Peter Rossmanith, *Worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT*, Discrete Appl. Math. **130** (2003), no. 2, 139–155. MR **MR2014655 (2004j:**68077)

[KF02] A. S. Kulikov and S. S. Fedin, *Solution of the maximum cut problem in time* $2^{|E|/4}$, Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI) **293** (2002), no. Teor. Slozhn. Vychisl. 7, 129–138, 183.

[Sco05] Alex Scott, *Judicious partitions and related problems*, Surveys in combinatorics 2005, London Math. Soc. Lecture Note Ser., vol. 327, Cambridge Univ. Press, Cambridge, 2005, pp. 95–117. MR MR2187736

[SS03] Alexander D. Scott and Gregory B. Sorkin, *Faster algorithms for MAX CUT and MAX CSP, with polynomial expected time for sparse instances*, Proc. 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003, Lecture Notes in Comput. Sci., vol. 2764, Springer, August 2003, pp. 382–395.

[SS04] ———, *A faster exponential-time algorithm for Max 2-Sat, Max Cut, and Max k-Cut*, Tech. Report RC23456 (W0412-001), IBM Research Report, December 2004, See http://domino.research.ibm.com/library/cyberdig.nsf/papers/3C1EC93E621268FA85256F6B005ED2AA/$File/rc23457.pdf.

[SS06] ———, *Solving sparse random instances of Max Cut and Max 2-CSP in linear expected time*, Comb. Probab. Comput. **15** (2006), no. 1-2, 281–315.

[Wil04] Ryan Williams, *A new algorithm for optimal constraint satisfaction and its implications*, Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP), 2004.

(Alexander D. Scott) Mathematical Institute, University of Oxford, 24-29 St Giles', Oxford, OX1 3LB, UK

E-mail address: scott@maths.ox.ac.uk

(Gregory B. Sorkin) Department of Mathematical Sciences, IBM T.J. Watson Research Center, Yorktown Heights NY 10598, USA

E-mail address: sorkin@watson.ibm.com