Extended Formulation for CSP that is Compact for Instances of Bounded Treewidth*

Petr Kolman, Martin Koutecký

Department of Applied Mathematics,
Faculty of Mathematics and Physics,
Charles University in Prague,
Czech Republic
kolman@kam.mff.cuni.cz, koutecky@kam.mff.cuni.cz

Submitted: Aug 7, 2015; Accepted: Nov 17, 2015; Published: Nov 27, 2015
Mathematics Subject Classifications: 03D15; 52B12; 90C05

Abstract

In this paper we provide an extended formulation for the class of constraint satisfaction problems and prove that its size is polynomial for instances whose constraint graph has bounded treewidth. This implies new upper bounds on extension complexity of several important NP-hard problems on graphs of bounded treewidth.

1 Introduction

Many important combinatorial optimization problems belong to the class of constraint satisfaction problems (CSP). Naturally, a lot of effort has been given to design efficient approximation algorithms for CSP, to prove complexity lower bounds for CSP, and to identify tractable instances of CSP (e.g., from the point of view of parameterized complexity). It has been shown that CSP is solvable in polynomial time for instances whose constraint graph has bounded treewidth [8].

In recent years, a lot of attention has been given to study *extension complexity* of problems [6]: given a problem Q, what is the minimum number of inequalities representing a polytope whose (suitably chosen) linear projection coincides with the convex hull H of all integral solutions of Q? Any polytope which projects to H is called an *extended formulation* of H. Note that membership of a problem in the class P of polynomially solvable problems does not necessarily imply the existence of an extended formulation of polynomial size [18]. In this work, we present an extended formulation for CSP and show that its size is polynomial for instances of CSP whose constraint graph has bounded treewidth.

*This research was partially supported by the project 14-10003S of GA ČR.

1.1 Notation and Terminology

An instance $Q = (V, \mathcal{D}, \mathcal{H}, \mathcal{C})$ of CSP consists of

- a set of *variables* z_v, one for each $v \in V$; without loss of generality we assume that $V = \{1, \dots, n\}$,
- a set $\mathcal{D}$ of finite *domains* $D_v \subseteq \mathbb{R}$ (also denoted $D(v)$), one for each $v \in V$,
- a set of *hard constraints* $\mathcal{H} \subseteq \{C_U \mid U \subseteq V\}$ where each hard constraint $C_U \in \mathcal{H}$ with $U = \{i_1, i_2, \dots, i_k\}$ and $i_1 < i_2 < \dots < i_k$, is a $|U|$-ary relation $C_U \subseteq D_{i_1} \times D_{i_2} \times \dots \times D_{i_k}$,
- a set of *soft constraints* $\mathcal{C} \subseteq \{C_U \mid U \subseteq V\}$ where each soft constraint $C_U \in \mathcal{C}$ with $U = \{i_1, i_2, \dots, i_k\}$ and $i_1 < i_2 < \dots < i_k$, is a $|U|$-ary relation $C_U \subseteq D_{i_1} \times D_{i_2} \times \dots \times D_{i_k}$.

The *constraint graph* of Q is defined as $G = (V, E)$ where $E = \{\{u, v\} \mid \exists C_U \in \mathcal{C} \cup \mathcal{H} \text{ s.t. } \{u, v\} \subseteq U\}$. We say that a *CSP instance Q has bounded treewidth* if the constraint graph of Q has bounded treewidth. In *binary CSP*, every hard and soft constraint is a unary or binary relation, and in *boolean CSP*, the domain D_v of every variable $v \in V$ is $D_v = \{0, 1\}$. We use D to denote the maximal size of all domains, that is, $D = \max_{u \in V} |D_u|$.

For a vector $z = (z_1, z_2, \dots, z_n)$ and a set $U = \{i_1, i_2, \dots, i_k\} \subseteq V$ with $i_1 < i_2 < \dots < i_k$, we define the *projection of z* on U as $z|_U = (z_{i_1}, z_{i_2}, \dots, z_{i_k})$. A vector $z \in \mathbb{R}^n$ *satisfies the constraint* $C_U \in \mathcal{C} \cup \mathcal{H}$ if and only if $z|_U \in C_U$. We say that a vector $z^\star = (z^\star_1, \dots, z^\star_n)$ is *a feasible assignment* for Q if $z^\star \in D_1 \times D_2 \times \ldots \times D_n$ and $z^\star$ satisfies every hard constraint $C \in \mathcal{H}$. For a given feasible assignment $z^\star$ we define an *extended feasible assignment* $\mathrm{ex}(z^\star) = (z^\star, h^\star) \in \mathbb{R}^{n+|\mathcal{C}|}$ as follows: the coordinates of $h^\star$ are indexed by the soft constraints from $\mathcal{C}$ (to be more precise: by the subsets U of V used as lower indices of the soft constraints) and for each $C_U \in \mathcal{C}$, we have $h^\star_U = 1$ if and only if $z^\star|_U \in C_U$, and $h^\star_U = 0$ otherwise. We denote by $\mathcal{F}(Q)$ the set of all feasible assignments for Q, by $\mathcal{F}^{ex}(Q) = \{\mathrm{ex}(z^\star) \mid z^\star \in \mathcal{F}(Q)\}$ the set of all extended feasible assignments for Q. For every instance Q we define two polytopes: $CSP(Q)$ is the convex hull of $\mathcal{F}^{ex}(Q)$ and $CSP'(Q)$ the convex hull of $\mathcal{F}(Q)$. We also define three trivial linear projections:

- $\mathrm{proj}_V(z, h) = z, \qquad \mathrm{proj}_E(z, h) = h, \qquad \mathrm{proj}_{id}(z, h) = (z, h)$

where $z \in \mathbb{R}^n$ and $h \in \mathbb{R}^{|\mathcal{C}|}$, and observe that $\mathrm{proj}_V(CSP(Q)) = CSP'(Q)$.

In the *decision* version of CSP, the set $\mathcal{C}$ of soft constraints is empty and the task is to decide whether there exists a feasible assignment. In the *maximization* (*minimization*, resp.) version of the problem, the task is to find a feasible assignment that maximizes (minimizes, resp.) the number of *satisfied* (unsatisfied, resp.) soft constraints. Note that there is no difference between maximization and minimization versions of the problem with respect to optimal solutions but the two versions differ significantly from an approximation perspective.

In the *weighted* version of CSP we are also given a weight function $w : \mathcal{C} \to \mathbb{R}$ that specifies for each soft constraint $C \in \mathcal{C}$ its weight $w(C)$. The goal is to find a feasible assignment that maximizes (minimizes, resp.) the total weight of satisfied (unsatisfied, resp.) constraints. The unweighted version of CSP is equivalent to the weighted version with $w(C) = 1$ for all $C \in \mathcal{C}$.

Even more generally, the relations in the soft constraints can be replaced by bounded real valued payoff functions: a soft constraint $C_U \in \mathcal{C}$ with $U = \{i_1, i_2, \ldots, i_k\}$ is not a $|U|$-ary relation but a function $w : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \to \mathbb{R}$ and the payoff of the soft constraint C_U for a feasible assignment $z^\star$ is $w(z^\star|_U)$; the objective is to maximize (minimize, resp.) the total payoff. For the sake of simplicity of the presentation we do not consider the problem in this generality although the techniques used in this paper apply in the general setting as well.

For notions related to the treewidth of a graph, we stick to the standard terminology as given in the book by Kloks [11]. For the sake of completeness, below we provide the definition of the tree decomposition and the nice tree decomposition of a graph.

A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (V_T, E_T)$ in which each *node* $a \in T$ has an assigned set of vertices $B(a) \subseteq V$ (called a *bag*) such that $\bigcup_{a \in V_T} B(a) = V$ with the following properties:

- for any $uv \in E$, there exists a node $a \in V_T$ such that $u, v \in B(a)$.
- if $v \in B(a)$ and $v \in B(b)$, then $v \in B(c)$ for all c on the path from a to b in T.

The *treewidth $tw(T)$ of a tree decomposition* T is the size of the largest bag of T minus one. The *treewidth $tw(G)$ of a graph* G is the minimum treewidth over all possible tree decompositions of G.

A *nice tree decomposition* is a tree decomposition with one special node r called the *root* in which each node is one of the following types:

- *Leaf node*: a leaf a of T with $|B(a)| = 1$.
- *Introduce node*: an internal node a of T with one child b for which $B(a) = B(b) \cup \{v\}$ for some $v \in B(a) \setminus B(b)$.
- *Forget node*: an internal node a of T with one child b for which $B(a) = B(b) \setminus \{v\}$ for some $v \in B(b)$.
- *Join node*: an internal node a with two children b and c with $B(a) = B(b) = B(c)$.

Note that restricting ourselves to nice tree decompositions is not a problem: any tree decomposition with b bags can be transformed in linear time into a nice one of the same width and with at most $4b$ bags [11].

1.2 Related Work

1.2.1 CSP for graphs of bounded treewidth.

As CSP captures many NP-hard problems, it is a natural problem to identify tractable special cases of CSP. Freuder [8] showed that CSP instances with treewidth bounded by τ can be solved in time $O(D^{\tau}n)$ when a tree decomposition of treewidth τ of the constraint graph is given. Later, Grohe et al. [9] proved that, assuming $FPT \neq W[1]$, this is essentially the only nontrivial class of graphs for which CSP is solvable in polynomial time (cf. Marx [14]).

Describing the polytope of CSP solutions by the means of linear programming, for instances of bounded treewidth, is not a new idea. In 2007, Sellmann et al. published a paper [21] in which they described a linear program that was supposed to define the convex hull of all feasible solutions of a binary CSP when the constraint graph is a tree. They also provided a procedure to convert a given CSP instance with bounded treewidth into one whose constraint graph is a tree, at the cost of blowing up the number of variables and constraints by a function of the treewidth. Unfortunately, there was a substantial bug in their proof and one of the main theorems in the paper does not even hold [20].

The paper [21] also implicitly includes this folklore result: if the constraint graph has treewidth at most τ, then CSP can be solved by τ levels of the Sherali-Adams hierarchy (we are not aware of an explicit formulation of this result in its generality though partial results of this type are known, e.g., for independent set [3]). The resulting formulation is of size $\mathcal{O}(n^{\tau})$ while our approach yields size $\mathcal{O}(D^{\tau}n)$.

1.2.2 CSP for general graphs.

Chan et al. [5] study the extent to which linear programming relaxation can be used in dealing with approximating CSP. They show that polynomial-sized LPs are exactly as powerful as LPs obtained from a constant number of rounds of the Sherali-Adams hierarchy. They also prove integrality gaps for polynomial-sized LPs for some CSP.

Raghavendra [15] shows that under the Unique Games Conjecture, a certain simple SDP relaxation achieves the best approximation ratio for every CSP. In a follow up paper, Raghavendra and Steurer [16] describe an efficient rounding scheme that achieves the integrality gap of the simple SDP relaxation, and, in another paper [17], they show unconditionally that the integrality gap of this SDP relaxation cannot be reduced by Sherali-Adams hierarchies.

1.2.3 Other related results.

Laurent [12] provides extended formulations for the independent set and the max cut problems, both a special case of CSP, that have size $O(2^{\tau}n)$ where τ denotes the treewidth of the given graph. These results are given in the context of moment matrices as an application of a sparsity structure present in instances of bounded treewidth. Independently, Buchanan and Butenko [4] later gave the same result for the independent set problem.

Our results can be viewed as a generalization: the size of our formulation for a general CSP, when applied to the independent set and the max cut problems, is also $O(2^{\tau}n)$.

In a recent work, independently of our result, Bienstock and Munoz [2] define a class of so called *general binary* optimization problems which are essentially weighted boolean CSP problems, and, among other results, for instances of treewidth τ provide an LP formulation of size $O(2^{\tau}n)$. Again, this is a special case of our result in this paper. It is worth mentioning at this point that every CSP instance can be transformed into a boolean CSP instance by encoding every variable with domain size D by $\lceil \log D \rceil$ boolean variables. This increases the treewidth of the constraint graph by a factor of $\lceil \log D \rceil$ and thus leads to a formulation of size $\mathcal{O}(2^{\tau \lceil \log D \rceil} n)$. This bound corresponds to $\mathcal{O}(D^{\tau}n)$ in the special case that D is a power of two, and to $\mathcal{O}(D^{\tau}2^{\tau}n)$ in the general case.

1.3 New Results

Our main result is summarized as the following theorem.

Theorem 1.1 *For every instance $Q = (V, \mathcal{D}, \mathcal{H}, \mathcal{C})$ of CSP, there exists an extended formulation $P(Q)$ of $CSP(Q)$ and $CSP'(Q)$ of size $\mathcal{O}(D^{\tau}n)$ where τ is the treewidth of Q; moreover, given a tree decomposition of the constraint graph of Q of width τ, the corresponding LP can be constructed in time $\mathcal{O}(D^{\tau}n)$.*

As a corollary we obtain upper bounds on the extension complexity for several NP-hard problems on the class of graphs with bounded treewidth; as far as we know, these results have not been known.

2 CSP Polytope

2.1 Integer Linear Programming Formulation

We start by introducing the terms and notation that we use throughout this section. We assume that $Q = (V, \mathcal{D}, \mathcal{H}, \mathcal{C})$ is a given instance of CSP. Let λ be a symbol not appearing in any of the domains D_u, $u \in V$. For every subset $W \subseteq V$ we define the set of all *configurations of* W as

$$\mathcal{K}(W) = \{(\alpha_1, \ldots, \alpha_n) \mid \forall C_U \in \mathcal{H} \ (U \subseteq W \implies \alpha|_U \in C_U), \text{ and } \forall i \notin W \ \alpha_i = \lambda\}$$

For a configuration $K \in \mathcal{K}(U)$ and $v \in V$, we use the notation $K(v)$ to refer to the v-th element of K. Also, for a configuration $K \in \mathcal{K}(U)$, $v \in V \setminus U$ and $\alpha \in D_v$, we use the notation $K[v \leftarrow \alpha]$ to denote the vector K' such that $K'(v) = \alpha$ and $K'(u) = K(u)$ for every $u \neq v$. Note that $K[v \leftarrow \alpha]$ does not necessarily have to be a configuration.

For an n-dimensional vector $K = (\alpha_1, \ldots, \alpha_n)$ and a subset of variables $U \subseteq V$ we denote by $K\restriction_U$ the *restriction of* K *to* U that is defined as an n-dimensional vector with $K\restriction_U(i) = K(i)$ for $i \in U$ and $K\restriction_U(i) = \lambda$ for $i \notin U$ (i.e., we set to λ all coordinates of K outside of U). We denote by Λ the configuration $(\lambda, \ldots, \lambda) \in \mathcal{K}(\emptyset)$; note that for $\alpha \in D_v$,

$\Lambda[v \leftarrow \alpha]$ is the vector with exactly one non-λ element, namely the v-th element, equaling α.

In our linear program, for every index $v \in V$ and every $i \in D_v$, we introduce a binary variable y_v^i. The task of the variable y_v^i is to encode the value of the CSP-variable z_v: the variable y_v^i is set to one if and only if $z_v = i$. Since in every solution each variable assumes a unique value, we enforce the constraint $\sum_{i \in D(v)} y_v^i = 1$ for each $v \in V$.

For every configuration $K \in \bigcup_{U: C_U \in \mathcal{C} \cup \mathcal{H}} \mathcal{K}(U)$ we introduce a binary variable $g(K)$. The intended meaning of the variable $g(K)$, for $K \in \mathcal{K}(U)$ and $U \subseteq V$, is to provide information about the values of the CSP-variables z_u for $u \in U$ in the following way: $g(K) = 1$ if and only if for every $u \in U$, $z_u = K(u)$. To ensure consistency between the y and g variables, for every $C_U \in \mathcal{C} \cup \mathcal{H}$ and for every $v \in U$, we enforce the constraint $\sum_{K \in \mathcal{K}(U): K(v)=i} g(K) = y_v^i$. Note that for binary CSP, the g variables capture the values of CSP-variables z for pairs of elements from V that correspond to edges of the constraint graph.

Relaxing the integrality constraints we obtain the following initial LP relaxation of the CSP problem $Q = (V, \mathcal{D}, \mathcal{H}, \mathcal{C})$:

$$\sum_{i \in D(v)} y_v^i = 1 \qquad \forall v \in V \tag{1}$$

$$\sum_{K \in \mathcal{K}(U): K(v)=i} g(K) = y_v^i \qquad \forall C_U \in \mathcal{C} \cup \mathcal{H} \ \forall v \in U \ \forall i \in D(v) \tag{2}$$

$$0 \leqslant \boldsymbol{y}, \boldsymbol{g} \leqslant 1. \tag{3}$$

Note that there is a one to one correspondence between the (extended) feasible assignments of Q and integral solutions of (1) - (3); from now on we denote by proj_1 the linear projection of the convex hull of integral solutions of (1) - (3) to $CSP(Q)$. Also observe that the total weight of CSP-constraints satisfied by an integral vector $(\boldsymbol{y}, \boldsymbol{g})$ satisfying (1) - (3) is[1]

$$\sum_{C_U \in \mathcal{C}} w(C_U) \sum_{K \in \mathcal{K}(U): K|_U \in C_U} g(K).$$

Unfortunately, even for CSP problems whose constraint graph is series-parallel, the polytope given by the LP (1) - (3) is not integral (consider, e.g., the instance of CSP corresponding to the independent set problem on K_3). The weakness of the formulation is that no *global* consistency among the $\boldsymbol{y}$ variables is guaranteed. To strengthen the relaxation, we introduce new variables and constraints derived from a tree decomposition of the constraint graph of Q.

[1]In the case of general payoff functions, the total weight is given by $\sum_{C_U \in \mathcal{C}} \sum_{K \in \mathcal{K}(U): K|_U \in C_U} w(K|_U) g(K)$

2.2 Extended Formulation

Here we describe, for every CSP instance $Q = (V, \mathcal{D}, \mathcal{H}, \mathcal{C})$, a polytope $P(Q)$, and in the next subsection we prove that $P(Q)$ is an extended formulation of $CSP(Q)$ and $CSP'(Q)$. The set of variables in the given LP description of $P(Q)$ is substantially different from the set of variables used in the LP (1) - (3), and the set of new constraints is completely different from the the set of constraints in the LP (1) - (3). Whereas in the previous subsection, there is (roughly) a variable $g(K)$ for every feasible assignment of every subset of CSP variables corresponding to a *soft or hard constraint*, here we have a variable for every feasible assignment of every subset of CSP variables corresponding to a *bag in a given tree decomposition of the constraint graph.* Nevertheless, as we show after defining $P(Q)$, there exists a simple linear projection of $P(Q)$ to the convex hull of all integral points in the polytope given by the LP (1) - (3).

Let $T = (V_T, E_T)$ be a fixed nice tree decomposition [11] of the constraint graph of Q and for every node $a \in V_T$, let $B(a) \subseteq V$ denote the corresponding bag. Let $\mathcal{B} = \{B(a) \mid a \in V_T\}$ denote the set of all bags of T. Let $\mathcal{K}_\mathcal{B} = \bigcup_{B \in \mathcal{B}} \mathcal{K}(B)$ be the set of all configurations of all bags in T. We use $V_I \subseteq V_T$ to denote the subset of all introduce nodes in T and $V_F \subseteq V_T$ to denote the subset of all forget nodes in T.

For every configuration $K \in \mathcal{K}_\mathcal{B}$ we introduce a binary variable $f(K)$. As in the previous subsection, the intended meaning of the variable $f(K)$ for $B \in \mathcal{B}$ and $K \in \mathcal{K}(B)$, is to provide information about the values of the CSP-variables z_u for $u \in B$ in the following way: $f(K) = 1$ if and only if for every $u \in B$, $z_u = K(u)$. To ensure consistency among variables indexed by the configurations of the same bag, namely to ensure that for every $B \in \mathcal{B}$ there exists exactly one configuration $K \in \mathcal{K}(B)$ with $f(K) = 1$, we introduce for every $B \in \mathcal{B}$ the LP constraint $\sum_{K \in \mathcal{K}(B)} f(K) = 1$.

For every introduce node $c \in V_T$ with a child $b \in V_T$ and for every configuration $K \in \mathcal{K}(B(b))$ we have the constraint $\sum_{K' \in \mathcal{K}(B(c)) : K' \restriction_{B(b)} = K} f(K') = f(K)$, and symmetrically, for every forget node $c \in V_T$ with a child $b \in V_T$ and for every configuration $K \in \mathcal{K}(B(c))$ we have the constraint $\sum_{K' \in \mathcal{K}(B(b)) : K' \restriction_{B(c)} = K} f(K') = f(K)$.

Relaxing the integrality constraints and putting all these additional constraints together, we obtain:

$$\sum_{K \in \mathcal{K}(B)} f(K) = 1 \qquad \forall B \in \mathcal{B} \tag{4}$$

$$\sum_{K' \in \mathcal{K}(B(c)) : K' \restriction_{B(b)} = K} f(K') = f(K) \qquad \forall c \in V_I, \forall K \in \mathcal{K}(B(b)) \text{ where } b \text{ is the only child of } c \tag{5}$$

$$\sum_{K' \in \mathcal{K}(B(b)) : K' \restriction_{B(c)} = K} f(K') = f(K) \qquad \forall c \in V_F, \forall K \in \mathcal{K}(B(c)) \text{ where } b \text{ is the only child of } c \tag{6}$$

$$0 \leqslant \boldsymbol{f} \leqslant 1. \tag{7}$$

For the given binary CSP instance Q, we denote the polytope associated with the LP (4) - (7), as $P(Q)$.

Consider now a vector $\boldsymbol{f} \in P(Q)$ and the following set of linear equations:

$$y_v^i = \sum_{K \in \mathcal{K}(B): K(v)=i} f(K) \qquad \forall B \in \mathcal{B}, \forall v \in B, \forall i \in D_v \tag{8}$$

$$g(K) = \sum_{K' \in \mathcal{K}(B): K' \restriction_U = K} f(K') \qquad \forall B \in \mathcal{B}, \forall C_U \in \mathcal{C} \cup \mathcal{H} \text{ s.t. } U \subseteq B, \forall K \in \mathcal{B}(U). \tag{9}$$

It is just a technical exercise to check that for a given $\boldsymbol{f} \in P(Q)$, there always exists a unique solution $(\boldsymbol{y}, \boldsymbol{g})$ of this LP and that the unique $(\boldsymbol{y}, \boldsymbol{g})$ is a linear projection of $\boldsymbol{f}$. Moreover, such a vector $(\boldsymbol{y}, \boldsymbol{g})$ also satisfies the LP constraints (1) - (3). The point is that there exists a linear projection, obtained from (8) - (9), of $P(Q)$ into the polytope defined by the LP (1) - (3); moreover, an integral point from $P(Q)$ is mapped on an integral point. From now on we denote this projection proj_2.

2.3 Proof of Theorem 1.1

As in the previous subsections, we assume that $Q = (V, \mathcal{D}, \mathcal{H}, \mathcal{C})$ is a given instance of CSP, $G = (V, E)$ is the constraint graph of Q and $T = (V_T, E_T)$ is a fixed nice tree decomposition of G. We start by introducing several notions that will help us dealing with tree decompositions and our linear program.

For a node $a \in V_T$, let $T(a) = (V_a, E_a)$ be the subtree of T rooted in a; the *configurations relevant to* $T(a)$ are those in the set $\mathcal{R}(a) = \bigcup_{b \in V_a} \mathcal{K}(B(b))$, and the *variables relevant to* $T(a)$ are those $f(K)$ for which $K \in \mathcal{R}(a)$. For succinctness of notation, we denote the projection $\boldsymbol{f}|_{\mathcal{R}(a)}$ of the vector $\boldsymbol{f}$ on the set of variables relevant to $T(a)$ also by $\boldsymbol{f}|_a$. The *constraints relevant to* $T(a)$ are those containing only the variables relevant to $T(a)$. We say that a vector $I \in \{0,1\}^{\mathcal{R}(a)}$ *agrees with the configuration* $K \in \mathcal{R}(a)$ if $I(K) = 1$.

Let $\boldsymbol{f}$ be a fixed solution of the LP (4) - (7) that corresponds to a vertex of the polytope $P(Q)$. Our main tool is the following lemma; our approach was partially inspired by the proof of the matching polytope theorem as given by Schrijver [19].

Lemma 2.1 *For every node $b \in V_T$, there exist a positive integer M and binary vectors $I_1, I_2, \ldots, I_M \in \{0,1\}^{\mathcal{R}(b)}$, some possibly identical, such that*

- ♠ *every I_i satisfies the constraints relevant to $T(b)$,*
- ♣ $\boldsymbol{f}|_b = \frac{1}{M} \sum_{i=1}^{M} I_i$.

Proof. By induction. We start in the leaves of T and proceed in a bottom-up fashion.

2.3.1 Base case.

Assume that $b \in V_T$ is a leaf of the nice decomposition tree T. By definition of a nice tree decomposition, the bag $B(b)$ consists of a single vertex, say a vertex $v \in V$. The only variables relevant to $T(b)$ are $f(K)$ for all $K \in \mathcal{K}(B(b)) = \{\Lambda[v \leftarrow j] \mid j \in D(v)\}$ and the only relevant constraints are those of the type (4) and (7).

Let $M' \in \mathbb{N}$ be such that an M'-multiple of every relevant variable is integral; as $\boldsymbol{f}$ is a solution corresponding to a vertex of the polytope $P(Q)$, all the variables are rational which guarantees that such an M' exists. For every $j \in D_v$ we define an integral vector I_j such that $I_j(\Lambda[v \leftarrow j]) = 1$ and $I_j(\Lambda[v \leftarrow i]) = 0$ for every $i \neq j$.

The vector I_j will appear with multiplicity $M' \cdot y_v^j$ among the integral solutions $I_1, \dots, I_{M'}$ for G'. Then, obviously, both properties $\spadesuit$ and $\clubsuit$ are satisfied.

2.3.2 Inductive step.

Consider an internal node $c \in V_T$ of the nice decomposition tree T. We distinguish three cases: c is a join node, c is an introduce node and c is a forget node.

Join node. Assume that the two children of the join node c are a and b. Recall that $B(a) = B(b) = B(c)$. By the inductive assumption, there exist integers M and M' and integral vectors $I_1, \dots, I_M \in \{0,1\}^{\mathcal{R}(a)}$, each of them satisfying the relevant constraints for $T(a)$ and such that $\boldsymbol{f}|_a = \frac{1}{M}\sum_{i=1}^{M} I_i$, and integral vectors $J_1, \dots, J_{M'} \in \{0,1\}^{\mathcal{R}(b)}$, each of them satisfying the relevant constraints for $T(b)$ and such that $\boldsymbol{f}|_b = \frac{1}{M'}\sum_{i=1}^{M'} J_i$.

Two vectors I_i and J_j that agree with a given configuration $K \in \mathcal{K}(B(c))$ can be easily merged into an integral vector $L \in \{0,1\}^{\mathcal{R}(c)}$ that satisfies $L|_a = I_i$ and $L|_b = J_j$; as the set of all constraints relevant to $T(c)$ is the union of the constraints relevant to $T(a)$ and the constraints relevant to $T(b)$, the vector L satisfies also all the constraints relevant to $T(c)$.

For simplicity we assume, without loss of generality, that $M = M'$. Then, by the property $\clubsuit$ and since $B(a) = B(b) = B(c)$, for every configuration $K \in \mathcal{K}(B(c))$, the number of vectors I_i that agree with K is equal to the number of vectors J_j that agree with K, namely $M \cdot f(K)$. Thus, it is possible to match the vectors I_i and J_j one to one in such a way that both vectors in each pair agree with the same configuration; let $L_1, L_2, \dots, L_M$ denote the result of their merging as described above. Then the vectors L_i satisfy the property $\spadesuit$ as explained in the previous paragraph, and by construction they also satisfy the property $\clubsuit$.

Introduce node. Assume that the only child of the introduce node c is a node b and $B(c) = B(b) \cup \{v\}$. By the inductive assumption, there exists integer M and integral vectors $I_1, \dots, I_M \in \{0,1\}^{\mathcal{R}(b)}$, each of them satisfying the relevant constraints for $T(b)$ and such that $\boldsymbol{f}|_b = \frac{1}{M}\sum_{i=1}^{M} I_i$. Without loss of generality we assume that for every variable relevant to $T(c)$, its M-multiple is integral. We partition the vectors $I_1, \dots, I_M$ into several groups indexed by the configurations from $\mathcal{K}(B(b))$: the group Z_K, for $K \in \mathcal{K}(B(b))$, consists exactly of those vectors I_i that agree with K.

Consider a fixed configuration $K \in \mathcal{K}(B(b))$ and the corresponding group Z_K. Note that the size of this group is $M \cdot f(K)$. We further partition the group Z_K into at most $|D_v|$ subgroups $Z_{K'}$, where $K' = K[v \leftarrow j]$, for every $j \in D_v$ satisfying $K[v \leftarrow j] \in \mathcal{K}(B(c))$, in such a way that $Z_{K'}$ contains exactly $M \cdot f(K')$ vectors (it does not matter which ones); the LP constraint (5) makes this possible. Then, for every $j \in D_v$, we create from every vector $I \in Z_{K[v \leftarrow j]}$ a new integral vector J_I in the following way:

- for every $\bar{K} \in \mathcal{R}(b)$, $J_I(\bar{K}) = I(\bar{K})$; this guarantees $J_I|_b = I$,

- $J_I(K[v \leftarrow j]) = 1$,
- for every $i \in D_v$, $i \neq j$, $J_I(K[v \leftarrow i]) = 0$.

Obviously, the new vectors J_I satisfy all constraints relevant to $T(b)$, and it is easy to check that they satisfy all constraints relevant to $T(c)$ as well, given the definitions above. Moreover, the definitions above imply that the vectors J_I satisfy the property $\clubsuit$.

Forget node. Assume that the only child of the forget node c is a node b, $B(c) = B(b) \setminus \{v\}$. This case is symmetric to the previous one in that instead of splitting the groups Z_K into smaller groups $Z_{K'}$, we merge them into bigger $Z_{K'}$.

By the inductive assumption, there exists an integer M and integral vectors $I_1, \dots, I_M \in \{0,1\}^{\mathcal{R}(b)}$, each of them satisfying the relevant constraints for $T(b)$ and such that $\boldsymbol{f}|_b = \frac{1}{M}\sum_{i=1}^{M} I_i$. Without loss of generality we assume that for every variable relevant to $T(c)$, its M-multiple is integral. We partition the vectors $I_1, \dots, I_M$ into several groups indexed by the configurations from $\mathcal{K}(B(b))$: the group Z_K, for $K \in \mathcal{K}(B(b))$, consists exactly of those vectors I_i that agree with K. Note that the size of Z_K is $M \cdot f(K)$.

For every $K' \in \mathcal{K}(B(c))$ we create a bigger group $Z_{K'}$ by merging $|D_v|$ of the groups Z_K, namely those satisfying $K|_{B(c)} = K'$. By the LP constraint (6), the new group $Z_{K'}$ contains exactly $M \cdot f(K')$ vectors. For every $K' \in \mathcal{K}(B(c))$, we create from every vector $I \in Z_{K'}$ a new integral vector J_I in the following way:

- for every $\bar{K} \in \mathcal{R}(b)$, $J_I(\bar{K}) = I(\bar{K})$.

If $\mathcal{K}(B(c)) \subseteq \mathcal{R}(b)$, there is nothing more to do. Otherwise we further define

- $J_I(K') = 1$, and for every $\hat{K} \in \mathcal{K}(B(c))$, $\hat{K} \neq K'$, $J_I(\hat{K}) = 0$.

We have to check that the vectors J_I satisfy all constraints relevant to $T(c)$. The only possibly new constraints are those using variables $f(K')$ for $K' \in \mathcal{K}(B(c))$ and it is easily seen that they are satisfied, given the definitions above. Also, the definitions above imply that the vectors $J_{K'}$ satisfy the property $\clubsuit$. $\square$

By applying Lemma 2.1 to the whole tree T, that is, to the subtree rooted in the root of T, we immediately obtain that $\boldsymbol{f}$ is an integral vector, and, thus, also the corresponding vertex of $P(Q)$ is integral. As this holds for every vertex of $P(Q)$, we conclude that $P(Q)$ is an integral polytope.

Considering the notes at the ends of the previous two subsections, we also conclude that $CSP(Q) = \mathrm{proj}_1(\mathrm{proj}_2(P(Q))$ and $CSP'(Q) = \mathrm{proj}_V(CSP(Q))$.

To complete the proof of Theorem 1.1, we observe that the number of variables and constraints in the LP (4) - (7) is $\mathcal{O}(D^\tau n)$.

3 Applications

The purpose of this section is to make explicit the extension complexity upper bounds given in Theorem 1.1 for several well known graph problems. Note that in all considered cases the constraint graph obtained from our CSP formulation is identical with the input graph; specifically, this means the treewidth of the CSP instance is the treewidth of the input graph. This is not obvious: for example, a natural CSP formulation of the Minimum Dominating Set problem involves constraints over a neighborhood of a vertex and thus has a possibly unbounded treewidth.

We find it interesting that the attained extension complexity upper bounds almost meet the best possible, assuming Strong ETH, *time complexity* lower bounds, given by Lokshtanov et al. [13]. They show for several problems whose time complexity is upper bounded by $\mathcal{O}(c^\tau n^{\mathcal{O}(1)})$, that they cannot be solved in time $\mathcal{O}((c-\epsilon)^\tau n^{\mathcal{O}(1)})$, for any $\epsilon > 0$, unless SETH fails. The only exception in our list is the Multiway Cut problem where the corresponding lower bounds are not known. To state our results, we use for each problem the following template:

Problem name Projection Extension complexity Time complexity

Instance: ...

Solution: ...

CSP formulation: V, $\mathcal{D}$, $\mathcal{H}$, $\mathcal{C}$. CSP version: Decision / Max / Min

where *Projection* is the name of the linear projection that yields the natural polytope of the problem Q from the $CSP(Q)$ polytope (or from the $P(Q)$ polytope, in case of the OCT problem). We use the notation $[n] = \{1, \dots, n\}$.

Coloring / Chromatic Number [1] proj_V $\mathcal{O}(q^\tau n)$ $\mathcal{O}(q^\tau n)$

Instance: Graph $G = (V, E)$, set of colors $[q]$.

Solution: A coloring of G with q colors with no monochromatic edges.

CSP formulation: $V = [n]$, $D_v = [q]$ for every $v \in V$, $H_{uv} = \{(i,j) \mid i \in D_u, j \in D_v, i \neq j\}$ for every $uv \in E$, $\mathcal{C} = \emptyset$. Decision

Comment: Note that Chromatic Number $\chi(G)$ of G is always upper bounded by $\tau + 1$ since graphs of bounded treewidth are τ-degenerate [11] and τ-degenerate graphs are $(\tau+1)$-colorable [22]. Thus, if the goal is to determine $\chi(G)$, it suffice to find the smallest q such that $CSP(Q)$ is non-empty.

List-H-Coloring / List Homomorphism [7] proj_V $\mathcal{O}(L^\tau n)$ $\mathcal{O}(L^\tau n)$

Instance: Graph $G = (V, E)$, graph $H = (V_H, E_H)$ possibly containing loops, and for every vertex $v \in V$ a set $L(v) \subseteq V_H$.

Solution: A mapping $f : V \to V_H$ such that $\forall uv \in E$ it holds that $f(u)f(v) \in E_H$ and $f(v) \in L(v)$ for every $v \in V$.

CSP formulation: $V = [n]$, $D_v = L(v)$ for every $v \in V$, $H_{uv} = \{(i,j) \mid i \in D_u, j \in D_v, ij \in E_H\}$ for every $uv \in E$, $\mathcal{C} = \emptyset$. Decision

Comment: Note that the problems List Coloring, Precoloring Extension and H-Coloring (or Graph Homomorphism) are all special cases of this problem. The lower bound given by Lokshtanov et al. [13] applies to all of them since Coloring is a special case of each of them.

Unique Games [10] proj$_{id}$ $\mathcal{O}(t^\tau n)$ —

Instance: Graph $G = (V, E)$, an integer $t \in \mathbb{N}$, a permutation π_e of order t for every edge $e \in E$.

Solution: A mapping $\ell : V \to [t]$ such that the number of edges $uv \in E$ with $\pi_{uv}(\ell(u)) = \ell(v)$ is maximized.

CSP formulation: $V = [n]$, $D_v = [t]$ for every $v \in V$, $\mathcal{H} = \emptyset$, $C_{uv} = \{(i, \pi_{uv}(i)) \mid i \in D_u\}$ for every edge $uv \in E$. Max

Comment: The decision variant of this problem is not interesting as it is trivially solvable in polynomial time.

Min Multiway Cut [1] proj$_E$ $\mathcal{O}(t^\tau n)$ $\mathcal{O}(t^\tau n)$

Instance: Graph $G = (V, E)$, an integer $t \in \mathbb{N}$ and t vertices $s_1, \ldots, s_t \in V$.

Solution: A partition of V into sets $V_1, \ldots, V_t$ such that for every i we have $s_i \in V_i$ and the total number of edges between V_i and V_j for $i \neq j$ is minimized.

CSP formulation: $V = [n]$, $D_v = [t]$ for every $v \in V$, $\mathcal{H} = \emptyset$, $C_{uv} = \{(i, i) \mid i \in [t]\}$ for every edge $uv \in E$. Max

Comment: Setting $z_v = i$ models vertex v belonging to the set V_i. Not satisfying the constraint C_{uv} means that the edge uv belongs to the multiway cut.

Max Cut [1] proj$_E$ $\mathcal{O}(2^\tau n)$ $\mathcal{O}(2^\tau n)$

Instance: Graph $G = (V, E)$.

Solution: A partition of vertices into two sets V_1, V_2 such that the number of edges between V_1 and V_2 is maximized.

CSP formulation: $V = [n]$, $D_v = \{0, 1\}$ for every $v \in V$, $\mathcal{H} = \emptyset$, $C_{uv} = \{(1, 0), (0, 1)\}$ for every edge $uv \in E$. Max

Comment: The values $0, 1$ model the vertex belonging to the set V_1 or V_2. If we replace maximization by minimization, the problem becomes Edge Bipartization (aka Edge OCT) problem which is a parametric dual of Max Cut.

Min Vertex Cover [1] proj$_V$ $\mathcal{O}(2^\tau n)$ $\mathcal{O}(2^\tau n)$

Instance: Graph $G = (V, E)$.

Solution: A set of vertices $C \subseteq V$ of minimal size such that every edge contains a vertex $v \in C$ as at least one of its endpoints.

CSP formulation: $V = [n]$, $D_v = \{0, 1\}$ for every $v \in V$, $H_{uv} = \{(1, 1), (0, 1), (1, 0)\}$ for every edge $uv \in E$, $C_v = \{0\}$ for every vertex $v \in V$. Min

Comment: The values $1, 0$ model the vertex belonging to C or $V \setminus C$.

Max Independent Set [1] proj$_V$ $\mathcal{O}(2^\tau n)$ $\mathcal{O}(2^\tau n)$

Instance: Graph $G = (V, E)$.

SOLUTION: A set of vertices $C \subseteq V$ of maximal size such that no edge contains both its endpoints in C.

CSP FORMULATION: $V = [n]$, $D_v = \{0, 1\}$ for every $v \in V$, $H_{uv} = \{(0, 0), (0, 1), (1, 0)\}$ for every edge $uv \in E$, $C_v = \{1\}$ for every vertex $v \in V$. MAX

Comment: The values $1, 0$ model the vertex belonging to C or $V \setminus C$.

ODD CYCLE TRANSVERSAL [13] $\quad \mathrm{proj}_{OCT} \circ \mathrm{proj}_2 \quad \mathcal{O}(3^\tau n) \quad \mathcal{O}(3^\tau n)$

INSTANCE: Graph $G = (V, E)$.

SOLUTION: A subset of vertices $W \subseteq V$ of minimal size such that $G[V \setminus W]$ is a bipartite graph.

CSP FORMULATION: $V = [n]$, $D_v = \{0, 1, 2\}$ for every $v \in V$, $H_{uv} = \{0, 1, 2\}^2 \setminus \{(0, 0), (1, 1)\}$ for every edge $uv \in E$, $C_v = \{0, 1\}$ for every $v \in V$. MIN

Comment: The values $0, 1, 2$ model the vertex belonging to either the first or the second partite of a bipartite graph, or the deletion set W. Satisfying the constraint C_v corresponds to not putting v in the deletion set W. Also known as VERTEX BIPARTIZATION. The projection $\mathrm{proj}_{OCT} : P(Q) \to \{0, 1\}^V$ is defined as

$$\mathrm{proj}_{OCT}(y_1^0, y_1^1, y_1^2, y_2^0, y_2^1, y_2^2, \ldots, y_n^0, y_n^1, y_n^2, \boldsymbol{g}) = (y_1^2, y_2^2, \ldots, y_n^2).$$

4 Open problems

A natural research direction is to examine more closely the extension complexity for CSP and the specific graph problems on graphs with bounded treewidth, in particular, what are the best possible upper bounds?

Acknowledgments

The authors thank Hans Raj Tiwary and Jiří Sgall for stimulating discussions.

References

[1] G. Ausiello, P. Creczenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation; Combinatorial Optimization Problems and Their Approximability Properties.* Springer, 1999.

[2] D. Bienstock and G. Munoz. LP approximations to mixed-integer polynomial optimization problems, arXiv:1501.00288v14 July 2015.

[3] D. Bienstock and N. Özbay. Tree-width and the sherali-adams operator. *Discrete Optimization*, 1(1):13–21, 2004.

[4] A. Buchanan and S. Butenko. Tight extended formulations for independent set, 2014. Available on Optimization Online.

[5] S. O. Chan, J. R. Lee, P. Raghavendra, and D. Steurer. Approximate constraint satisfaction requires large LP relaxations. In *Proc. of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 350–359, 2013.

[6] M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *Annals OR*, 204(1):97–143, 2013.

[7] T. Feder and P. Hell. List homomorphisms to reflexive graphs. *J. Comb. Theory, Ser. B*, 72(2):236–250, 1998.

[8] E. C. Freuder. Complexity of K-tree structured constraint satisfaction problems. In *Proc. of the 8th National Conference on Artificial Intelligence*, pages 4–9, 1990.

[9] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Proc. of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 657–666, 2001.

[10] S. Khot. On the power of unique 2-Prover 1-Round games. In *Proc. of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 767–775, 2002.

[11] T. Kloks. *Treewidth: Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science.* Springer, 1994.

[12] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.

[13] D. Lokshtanov, D. Marx, and S. Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In *Proc. of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 777–789, 2011.

[14] D. Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010.

[15] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proc. of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 245–254, 2008.

[16] P. Raghavendra and D. Steurer. How to round any CSP. In *Proc. of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 586–594, 2009.

[17] P. Raghavendra and D. Steurer. Integrality gaps for strong SDP relaxations of unique games. In *Proc. of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 575–585, 2009.

[18] T. Rothvoß. The matching polytope has exponential extension complexity. In *Proc. of the 46th ACM Symposium on Theory of Computing (STOC)*, pages 263–272, 2014.

[19] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1986.

[20] M. Sellmann. The polytope of tree-structured binary constraint satisfaction problems. In *Proc. of Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, volume 5015 of *Lecture Notes in Computer Science*, pages 367–371. Springer, 2008.

[21] M. Sellmann, L. Mercier, and D. H. Leventhal. The linear programming polytope of binary constraint problems with bounded tree-width. In *Proc. of Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, volume 4510 of *Lecture Notes in Computer Science*, pages 275–287. Springer, 2007.

[22] G. Szekeres and H. S. Wilf. An inequality for the chromatic number of a graph. *Journal of Combinatorial Theory*, 4(1):1–3, 1968.