Linear programming, width-1 CSPs, and robust satisfaction

Gabor Kun* Ryan O'Donnell† Suguru Tamaki‡ Yuichi Yoshida§ Yuan Zhou¶

June 16, 2012

Abstract

We say that an algorithm *robustly decides* a constraint satisfaction problem Π if it distinguishes at-least-$(1-\epsilon)$-satisfiable instances from less-than-$(1-r(\epsilon))$-satisfiable instances for some function $r(\epsilon)$ with $r(\epsilon) \to 0$ as $\epsilon \to 0$. In this paper we show that the canonical linear programming relaxation robustly decides Π if and only if Π has "width 1" (in the sense of Feder and Vardi).

*Institute for Advanced Study & DIMACS.

†Department of Computer Science, Carnegie Mellon University. Supported by NSF grant CCF-0747250 and a Sloan fellowship. Research performed while the author was a von Neumann Fellow at the Institute for Advanced Study.

‡School of Informatics, Kyoto University.

§School of Informatics, Kyoto University, and Preferred Infrastructure, Inc. Supported by MSRA Fellowship 2010.

¶Department of Computer Science, Carnegie Mellon University.

1 Introduction

Constraint satisfaction problems (CSPs) constitute a broad and important subclass of algorithmic tasks. One approach to studying the complexity of CSPs centers around the Feder–Vardi *Dichotomy Conjecture* [9] and the use of algebra [13] to classify all CSP decision problems. Another approach to the study of CSPs involves quantifying the extent to which natural CSPs can be *approximately* solved [14]; this approach has been characterized by more "analytic" methods. Recently there has been interest in melding the two approaches (see, e.g., [18, 15, 11]); the present work takes another step in this direction.

Almost-satisfiable instances. The algebraic approach to CSPs is mainly concerned with what we'll call the *decision* problem for CSPs: given an instance, is it completely satisfiable? The Dichotomy Conjecture states that for every CSP this task is either in P or is NP-hard; the *Algebraic Dichotomy Conjecture* of Bulatov, Jeavons, and Krokhin [4] refines this by conjecturing a precise algebraic characterization of the tractable CSP decision problems. However when it comes to approximability, not all tractable CSPs are "equally tractable". E.g., for Max-Cut, not only can one efficiently find a completely satisfying assignment when one exists, the Goemans–Williamson algorithm [10] efficiently finds an *almost*-satisfying assignment whenever an *almost*-satisfying assignment exists. (Specifically, it finds a $(1 - O(\sqrt{\epsilon}))$-satisfying assignment whenever a $(1 - \epsilon)$-satisfying assignment exists.) Contrast this with the kLin(mod 2) problem, $k \geq 3$: again, one can efficiently find a completely satisfying assignment whenever one exists; however Håstad [12] has shown that finding even a *somewhat*-satisfying assignment whenever an *almost*-satisfying assignment exists is NP-hard. (Specifically, $\forall \epsilon > 0$ it is hard to find a $(1/2 + \epsilon)$-satisfying assignment when a $(1 - \epsilon)$-satisfying assignment exists.)

Prior work on robust decidability. In 1997, Zwick [22] initiated the study of the following very natural problem: which CSPs are efficiently *robustly decidable*? By this we mean that the algorithm should find $(1 - o_\epsilon(1))$-satisfying assignments whenever $(1 - \epsilon)$-satisfying assignments exist (formal definitions are given in Section 2). Zwick gave a linear programming (LP)-based algorithm for finding $(1 - O(1/\log(1/\epsilon)))$-satisfying assignments for Horn-kSat (for any fixed k); he also gave a semidefinite programming (SDP)-based algorithm for finding $(1 - O(\epsilon^{1/3}))$-satisfying assignments for 2Sat (since improved to $1 - O(\epsilon^{1/2})$ [7]). Later, Khot [17] gave an SDP-based algorithm for finding $(1 - \tilde{O}(\epsilon^{1/5}))$-satisfying assignments for the notorious *Unique-Games* problem over domains D with $|D| = O(1)$ (since improved to $1 - O(\epsilon^{1/2})$ [6]).[1] On the other hand, the only tractable CSPs for which the robust decision problem seems to be NP-hard are the ones that can encode linear equations over groups.

Bounded width. If we wish to classify the CSPs which are efficiently robustly decidable, we seek a property that is shared by Horn-kSat, 2Sat, and Unique-Games but not by 3Lin(mod p). From the algebraic viewpoint on CSPs there is a very obvious candidate: the former CSPs have *bounded width* while the latter does not. Briefly, a CSP is said to have *bounded width* if unsatisfiable instances can always be "refuted" in a proof system that only allows for *constant*-sized partial assignments to be kept "in memory" (again, more formal definitions are in Section 2). Recent independent works of Barto–Kozik [1] and Bulatov [3] have connected this notion to algebra by showing that bounded-width CSPs coincide with those which cannot encode linear equations over groups. Thus

[1] We emphasize that in this paper, we always treat the domain size $|D|$ as a fixed constant, with $\epsilon \to 0$ independently.

by Håstad's work we know that any CSP which is efficiently robustly decidable must have bounded width (assuming $\mathsf{P} \neq \mathsf{NP}$). A very appealing recent conjecture of Guruswami and Zhou [11] is that the converse also holds: every bounded-width CSP has an efficient robust decision algorithm.

Linear and semidefinite programming. Essentially the only known way to produce CSP approximation algorithms is through the use of LPs and SDPs. Indeed, recent work of Raghavendra [20] shows that if one believes Khot's Unique Games Conjecture [17], then a CSP Π is efficiently robustly decidable if and *only if* the basic SDP relaxation robustly decides it. However understanding and solving SDPs can be difficult, and as Zwick's Horn-kSat algorithm illustrates, sometimes only the power of linear programming is needed for robust decision algorithms. There is also a close connection between robust decidability by linear programming and the problem of *property testing* for CSPs for satisfiability; this is described in Appendix B.

1.1 Our contribution

As a step towards the Guruswami–Zhou conjecture, we completely resolve the question of which CSPs are robustly decidable by the basic *linear programming relaxation*. Somewhat informally stated, our main theorem is the following:

Theorem 1.1. *Let* Π *be any (finitely presented) CSP. Then the basic LP relaxation robustly decides* Π *if and only if* Π *has* width 1*.*

(Formal definitions of the terms in this theorem are given in Section 2, and we further discuss the notion of "width 1" below.)

"If" part of Theorem 1.1: the basic LP robustly decides width-1 CSPs. This is formally stated as Theorem 3.1 and is proved in Section 3. Our proof gives an efficient deterministic "LP-rounding" algorithm for actually *finding* the required almost-satisfying assignments. Quantitatively, it finds $(1 - O(1/\log(1/\epsilon)))$-satisfying assignments for $(1-\epsilon)$-satisfiable instances, matching the performance of Zwick's Horn-kSat algorithm. As we describe below, this is best possible. Our rounding algorithm is also simpler than Zwick's. (An alternate proof of Theorem 1.1 is given in Appendix D.)

Independently and concurrently, Dalmau and Krokhin have also shown that width 1 CSPs have efficient robust decision algorithms. Their proof is different from ours; it is by a black-box reduction to Zwick's Horn-Sat algorithm.

"Only if" part of Theorem 1.1: the basic LP robustly decides *only* width-1 CSPs. In fact we prove a stronger result, Theorem 4.1 in Section 4: the basic LP *exactly* decides satisfiability only for width-1 CSPs.

Width 1: characterizations. The class of "width-1" CSPs, introduced by Feder and Vardi [9], is a well-known and long-studied subclass of bounded-width CSPs. The canonical example is Horn-Sat; additional examples are mentioned in Appendix A. We define "width-1" CSPs precisely in Section 2, but briefly, they can be characterized as the CSPs for which satisfiability is correctly decided by the (generalized) *arc-consistency* algorithm [19, 8]. Informally, this is also equivalent to saying that unsatisfiable instances can be refuted while keeping just single variable assignments "in memory". Dalmau and Pearson [8] have also straightforwardly characterized width-1 CSPs as those possessing a "set operation"; this will also be defined in Section 2.

Our Theorem 1.1 gives new characterizations of width 1; indeed, by the end of this work we will have established the following:

Theorem 1.2. *For a CSP* Π*, the following are equivalent:*

1. Π *has width 1;*
2. Π *has tree duality;*
3. Π *has a set operation;*
4. Π *has a "measure operation";*
5. Π *has symmetric polymorphisms of every arity;*
6. *the basic LP relaxation decides satisfiability for* Π*;*
7. *the basic LP relaxation robustly decides satisfiability for* Π*;*
8. *satisfiability for* Π *is "property-testable in the bounded-degree model".*

The key to the "if" part of Theorem 1.1 is the implication (3)⇒(7). The key to the "only if" part of Theorem 1.1 is (4)⇒(2).[2] All other implications were already known or are relatively straightforward.

The quantitative dependence on ϵ. As mentioned, our LP-based algorithm for width-1 CSPs finds $(1 - O(1/\log(1/\epsilon)))$-satisfying assignments to $(1-\epsilon)$-satisfiable instances. One might hope for a better (say, polynomial) dependence on ϵ here. Unfortunately, this is not possible. Zwick [22] already showed that for Horn-3Sat there are "gap instances" where the basic LP has value $1-\epsilon$ but the optimum value is only $1 - \Omega(1/\log(1/\epsilon))$. Indeed Guruswami and Zhou [11] extended this by showing there are equally bad gap instances for the basic SDP relaxation of Horn-3Sat. Assuming the Unique Games Conjecture, Raghavendra's work [20] in turn implies that *no* polynomial-time algorithm can find $(1 - o(1/\log(1/\epsilon)))$-satisfying assignments to $(1-\epsilon)$-satisfiable instances. On a positive note, in Appendix A we show that for the special case of width-1 CSPs called "lattice CSPs", the basic LP relaxation can be used to find $(1 - O(\epsilon))$-satisfying assignments to $(1-\epsilon)$-satisfiable instances.

2 Preliminaries

2.1 CSP preliminaries

Definitions. Let D be a nonempty finite *domain* of values, and let Γ be a nonempty finite set of *relations* over D, each of positive finite arity. We write such a k-ary relation as $R : D^k \to \{0,1\}$. An *instance* $\mathcal{I}$ of the *constraint satisfaction problem* CSP(Γ) consists of a set V of n *variables*, along with a list of m *constraints*. Each constraint C is a pair (S, R), where S is a tuple of some k variables (the *scope* of the constraint), and R is a k-ary relation in the set Γ. We say that $\mathcal{I}'$ is a *sub-instance* of $\mathcal{I}$ if contains just a subset of the variables and constraints in $\mathcal{I}$; it is *induced* by the variable set $V' \subseteq V$ if it includes all constraints in $\mathcal{I}$ involving just the variables in V'. An *assignment* for an instance of CSP(Γ) is any mapping $\alpha : V \to D$. The assignment *satisfies* a constraint $C = (S, R)$ if $R(\alpha(S)) = 1$ (where α operates on S_i component-wise). The *value* of the assignment, $\mathrm{Val}_{\mathcal{I}}(\alpha) \in [0,1]$, is the fraction of constraints it satisfies. We define the *optimum value* of the instance $\mathcal{I}$ to be $\mathrm{Opt}(\mathcal{I}) = \max_\alpha\{\mathrm{Val}_{\mathcal{I}}(\alpha)\}$. We say the instance is *satisfiable* if $\mathrm{Opt}(\mathcal{I}) = 1$.

[2] The essentially similar result (5)⇒(2) was announced by the first author in [18]; the proof appears here for the first time.

CSP width. An important parameter of a CSP(Γ) problem is its *width*. This notion, dating back to Feder and Vardi [9], can be given many equivalent definitions (in terms of, e.g., pebble games, Datalog, logic, tree-width, proof complexity...). Roughly speaking, CSP(Γ) has width k if unsatisfiable instances of CSP(Γ) can always be refuted while only keeping k partial assignments "in memory". More formally, given an instance $\mathcal{I}$ of CSP(Γ), consider the following (k, ℓ) *pebble game* with $1 \leq k < \ell$ integers: Alice begins by placing each of ℓ pebbles on variables in V. Bob must respond with a partial assignment to the pebbled variables which satisfies all constraints in which they participate. On each subsequent turn, Alice may move $\ell - k$ of the pebbles to different vertices. Bob must respond with a partial assignment to the newly pebbled variables which again satisfies all constraints in which the pebbled variables participate, and which is consistent with the assignment to the k unmoved pebbles from the previous turn. If ever Bob cannot respond, Alice wins the game; if Bob can always play forever, he wins the game. If $\mathcal{I}$ is a satisfiable instance then Bob can always win regardless of k and ℓ; on the other hand, if $\mathcal{I}$ is unsatisfiable, then Alice may or may not be able to win. We say that CSP(Γ) has *width* (k, ℓ) if Alice can win the (k, ℓ) pebble game on all unsatisfiable instances; and, we say that CSP(Γ) has *width* k if it has width (k, ℓ) for some finite ℓ. In particular, we say that CSP(Γ) has *bounded width* if it has width k for some finite k. Bounded width CSPs can be solved in polynomial time using a simple enumeration over Bob's possible strategies. As examples, Horn-kSat has width 1, 2-Colorability has width 2 (but not width 1), and 3Lin(mod 2) does not have bounded width.

Tree duality and width 1. It is well known [9] that the CSPs of width 1 can be precisely characterized as those which have *tree duality*. We say that CSP(Γ) has tree duality if for every unsatisfiable instance $\mathcal{I}$ there is a unsatisfiable "tree" instance $\mathcal{T}$ which "witnesses" this. By "witness" we mean that there is a *homomorphism* from $\mathcal{T}$ to $\mathcal{I}$; i.e., a map from $\mathcal{T}$'s variables into $\mathcal{I}$'s variables which preserves all relations. The definition of a "tree" instance is the natural one in case all relations in Γ have arity 2; in general, we must make more careful definitions. We define a *walk* in instance $\mathcal{I}$ of CSP(Γ) to be a sequence $x_1, C_1 = (S_1, R_1), t_1, u_1, x_2, C_2 = (S_2, R_2), t_2, u_2, \ldots, x_{\ell+1}$ where each x_i is a variable in $\mathcal{I}$, each C_i is a constraint in $\mathcal{I}$, the indices t_i and u_i are distinct, and $(S_i)_{t_i} = x_i$, $(S_i)_{u_i} = x_{i+1}$ for all $i \in [\ell]$. We say the walk proceeds from *starting point* x_1 to *endpoint* x_ℓ. We say the walk is *non-backtracking* if for every $i \in [\ell]$ either C_i differs from C_{i+1} or $u_i \neq t_{i+1}$. We say that $\mathcal{I}$ is *connected* if there is a walk from x to y for all pairs of distinct variables x and y in $\mathcal{I}$. Finally, we say that $\mathcal{I}$ is a *tree* if it is connected and it does not contain any non-backtracking walk with the same starting point and endpoint.

2.2 Algorithmic preliminaries

Approximation algorithms. For real numbers $0 \leq s \leq c \leq 1$, we say an algorithm (c, s)-*approximates* CSP(Γ) if it outputs an assignment with value at least s on any input instance with value at least c. For $c = s = 1$ we simply say that the algorithm *decides* CSP(Γ); this means the algorithm always finds a satisfying assignment given a satisfiable instance. We say that an algorithm *robustly decides* CSP(Γ) if there is an *error function* $r : [0, 1] \to [0, 1]$ with $r(\epsilon) \to 0$ as $\epsilon \to 0$ such that the algorithm $(1 - \epsilon, 1 - r(\epsilon))$-approximates CSP($\Gamma$) for all $\epsilon \in [0, 1]$. In particular, the algorithm must decide CSP(Γ).

The basic integer program. For any instance $\mathcal{I}$ of CSP(Γ) there is an equivalent canonical 0-1 integer program we denote by IP($\mathcal{I}$). It has variables $p_v(j)$ for each $v \in V$, $j \in D$, as well as variables $q_{C_i}(J)$ for each arity-k constraint $C_i = (S_i, R_i)$ and tuple $J \in D^{k_i}$. The interpretation of

$p_v(j) = 1$ is that variable v is assigned value j; the interpretation of $q_{C_i}(J) = 1$ is that the k_i-tuple of variables S_i is assigned the k_i-tuple of values J. More formally, $\mathrm{IP}(\mathcal{I})$ is the following:

$$\text{maximize} \quad \frac{1}{m}\sum_{i=1}^{m} \sum_{J:R_i(J)=1} q_{C_i}(J)$$

$$\text{subject to:} \quad \sum_{j\in D} p_v(j) = 1 \text{ for all } v \in V, \tag{1}$$

$$\sum_{J\in D^{k_i}:J_t=j} q_{C_i}(J) = p_v(j) \text{ for all } C_i \text{ and } t \text{ such that } (S_i)_t = v. \tag{2}$$

The optimum value of $\mathrm{IP}(\mathcal{I})$ is precisely $\mathrm{Opt}(\mathcal{I})$. Note that the size of this integer programming formulation is $\mathrm{poly}(n, m)$ (as we are assuming that D and Γ are of constant size).

The basic linear program. If we relax $\mathrm{IP}(\mathcal{I})$ by having the variables take values in the range $[0, 1]$ rather than $\{0, 1\}$, we obtain the *basic linear programming relaxation* which we denote by $\mathrm{LP}(\mathcal{I})$. An optimal solution of $\mathrm{LP}(\mathcal{I})$ can be computed in $\mathrm{poly}(n, m)$ time; the optimal value, which we denote by $\mathrm{LPOpt}(\mathcal{I})$, always satisfies $\mathrm{Opt}(\mathcal{I}) \le \mathrm{LPOpt}(\mathcal{I}) \le 1$. We interpret any feasible solution to $\mathrm{LP}(\mathcal{I})$ as follows: For each $v \in V$, the quantities $p_v(j)$ form a discrete probability distribution on D (because of (1)), denoted p_v. For each k_i-ary constraint $C_i = (S_i, R_i)$, the quantities $q_{C_i}(J)$ form a probability distribution on D^{k_i}, denoted q_{C_i}. Furthermore (because of (2)), the marginals of the q_{C_i} distributions are "consistent" with the p_v distributions, in the sense that whenever $(S_i)_t = v$ it holds that $\mathbf{Pr}_{J\sim q_{C_i}}[J_t = j] = p_v(j)$ for all $j \in D$. Finally, the objective value to be optimized in $\mathrm{LP}(\mathcal{I})$ is

$$\mathrm{LPVal}_{\mathcal{I}}(\{p_v\}, \{q_{C_i}\}) = \frac{1}{m}\sum_{i=1}^{m} \mathop{\mathbf{Pr}}_{J\sim q_{C_i}}[R_i(J) = 1];$$

the optimum value of this over all feasible solutions is $\mathrm{LPOpt}(\mathcal{I})$.

2.3 Algebraic preliminaries

Polymorphisms. The Dichotomy Conjecture of Feder and Vardi [9] asserts that for each Γ, the problem of deciding $\mathrm{CSP}(\Gamma)$ is either in P or is NP-complete. The most successful approach towards this conjecture has been the algebraic one initiated by Jeavons and coauthors [13] in which the problem is studied through the *polymorphisms* of Γ. We say $f : D^\ell \to D$ is an ℓ-ary polymorphism for the k-ary relation R if $R(f(x^1), \ldots, f(x^k)) = 1$ whenever $R(x_i^1, \ldots, x_i^k) = 1$ for all $i \in [\ell]$ (here each x^j is a tuple in D^ℓ). We say that f is a polymorphism for Γ if it is a polymorphism for each relation in Γ. We say that Γ is a *core*, if all of its 1-ary polymorphisms are bijections (at a high level, this means that there are no superfluous values in D for $\mathrm{CSP}(\Gamma)$). Finally, we call a polymorphism f *idempotent*, if $f(j, \ldots, j) = j$ for all $j \in D$.

Polymorphisms and width. Recently, independent works of Barto–Kozik [1] and Bulatov [3] managed to characterize bounded-width CSPs in terms of their polymorphisms. Specifically, they showed that $\mathrm{CSP}(\Gamma)$ has bounded width (for Γ a core) if and only if Γ has an ℓ-ary *weak near-unanimity (WNU)* polymorphism for all $\ell \ge 3$. Here a polymorphism f is said to be WNU if

it is idempotent and has the following symmetry: $f(x,x,\ldots,x,y) = f(x,\ldots,x,y,x) = \cdots = f(x,y,x,\ldots,x) = f(y,x,x,\ldots,x)$.

Much earlier, Dalmau and Pearson [8] gave a straightforward characterization the class of width-1 CSPs in terms of their polymorphisms. Specifically, they showed that CSP(Γ) has width 1 if and only if Γ is preserved by a *set operation* $g : \mathcal{P}(D) \to D$. This means that $f : D^\ell \to D$ defined by $f(x_1,\ldots,x_\ell) = g(\{x_1,\ldots,x_\ell\})$ is a polymorphism for all $\ell \geq 1$. Note that all these polymorphisms are *symmetric*, meaning invariant under all permutations of the inputs. We will use a simple lemma about width-1 CSPs which first requires a definition.

Definition 2.1. Let $\mathcal{J}$ be a subset of a cartesian product $B_1 \times \cdots \times B_k$ of nonempty sets. We say $\mathcal{J}$ is *subdirect*, written $\mathcal{J} \subseteq_S B_1 \times \cdots \times B_k$, if for each $i \in [k]$ the projection of $\mathcal{J}$ to the i'th coordinate is all of B_i.

Lemma 2.2. *Say g is a set operation for* CSP(Γ)*, R is an arity-k relation in Γ, and $B_1,\ldots,B_k \subseteq D$. Assume there is a $\mathcal{J} \subseteq_S B_1 \times \cdots \times B_k$ all of whose members satisfy R. Then $R(g(B_1),\ldots,g(B_k)) = 1$.*

Proof. For each $t \in [k]$ and $j \in B_t$, select some $J^{t,j} \in \mathcal{J}$ whose t'th coordinate is j.. Think of the $\ell = \sum |B_t| \geq 1$ tuples $J^{t,j}$ as column vectors, and adjoin them in some order to form a $k \times \ell$ matrix X. Let x^t be the t^{th} row of X. It is clear that the set of values appearing in x^t is precisely B_t. Thus if f is the ℓ-ary polymorphism defined by g, we have $f(x^t) = g(B_t)$. But since f is a polymorphism and each $J^{t,j}$ satisfies R, it follows that $R(g(B_1),\ldots,g(B_k)) = 1$. □

3 Width 1 implies robust decidability by LP

The following theorem shows that a simple rounding algorithm for the basic linear program robustly decides any width-1 CSP.

Theorem 3.1. *Let Γ be a finite set of relations over the finite domain D, each relation having arity at most K. Assume that* CSP(Γ) *has width* 1*. Then there is a* poly(n,m)*-time algorithm for* CSP(Γ) *which when given an input $\mathcal{I}$ with* LPVal$(\mathcal{I}) = 1-\epsilon$ *outputs an assignment $\alpha : V \to D$ with* $\text{Val}_{\mathcal{I}}(\alpha) \geq 1 - O(K^2|D|\log(2|D|))/\log(1/\epsilon)$*. (In particular,* $\text{Val}_{\mathcal{I}}(\alpha) = 1$ *if* LPVal$(\mathcal{I}) = 1$*.)*

Proof. The first step of the algorithm is to solve the LP relaxation of the instance, determining an optimal solution $\{p_v : v \in V\}$, $\{q_{C_i} : i \in [m]\}$ which obtains LPVal$(\mathcal{I}) = 1-\epsilon$. For technical reasons we will now assume without loss of generality that $K \geq 2$ and that

$$2^{-\text{poly}(n,m)} \leq \epsilon \leq \frac{1}{4(2|D|)^{2(K-1)}}. \tag{3}$$

The assumption (3) is also without loss of generality. We may assume the upper bound by adjusting the constant in the $O(\cdot)$ of our theorem. As for the lower bound, since linear programming is in polynomial time, ϵ will be either 0 or at least $2^{-\text{poly}(n,m)}$. In the former case, we replace ϵ with a sufficiently small $2^{-O(m)}$ so that the theorem's claimed lower bound on $\text{Val}_{\mathcal{I}}(\alpha)$ exceeds $1-1/m$; then $\text{Val}_{\mathcal{I}}(\alpha) > 1-1/m$ implies $\text{Val}_{\mathcal{I}}(\alpha) = 1$ as required when LPVal$(\mathcal{I}) = 1$.

For a particular constraint C_i, let $\epsilon_i = \sum_{J:R_i(J)=0} q_{C_i}(J)$. Since LPVal$(\mathcal{I}) \geq 1-\epsilon$ we have $\text{avg}\{\epsilon_i\} \leq \epsilon$. The next step is to "give up" on any constraint having $\epsilon_i > \sqrt{\epsilon}$. By Markov's inequality the fraction of such constraints is at most $\sqrt{\epsilon}$, which is negligible compared to the $O(1/\log(1/\epsilon))$ error guarantee of our algorithm. For notational simplicity, we now assume that $\epsilon_i \leq \sqrt{\epsilon}$ for all $i \in [m]$.

We now come to the main part of the algorithm. Since CSP(Γ) has width 1, it has a set operation $g : \mathcal{P}(D) \to D$. We first describe a simple *randomized* "LP-rounding" algorithm based on g:

1. Let $r = (2|D|)^{K-1}$ and let $b = \lfloor \log_r(1/2\sqrt{\epsilon}) \rfloor$. We have $b \geq 1$ by (3).
2. Choose $\theta \in \{r^{-1}, r^{-2}, \ldots, r^{-b}\}$ uniformly at random. Note that $r^{-b} \geq 2\sqrt{\epsilon}$.
3. Output the assignment $\alpha : V \to D$ defined by $\alpha(v) = g(\mathrm{supp}_\theta(p_v))$, where $\mathrm{supp}_\theta(p_v)$ denotes $\{j \in D : p_v(j) \geq \theta\}$.

We will show for each constraint $C_i = (S_i, R_i)$ that

$$\mathbf{Pr}[R_i(\alpha(S_i)) = 0] \leq K|D|/b. \tag{4}$$

It follows from linearity of expectation that the expected fraction of constraints not satisfied by α is at most $K|D|/b = O(K^2|D|\log(2|D|))/\log(1/\epsilon)$. This would complete the proof, except for the fact that we have given a randomized algorithm. However we can easily make the algorithm deterministic and efficient by trying all choices for θ (of which there are at most $b \leq \mathrm{poly}(n, m)$ by (3)) and selecting the best resulting assignment.

We now give the analysis justifying (4) for each fixed constraint $C_i = (S_i, R_i)$. For simplicity we henceforth write $C = C_i$, $S = S_i$, $R = R_i$ and suppose that R has arity $k \leq K$. Let us say that a choice of θ is *bad* if it falls into the interval $(p_{S_t}(j)/r, p_{S_t}(j)]$ for some $t \in [k]$ and $j \in D$. For each choice of t and j there is at most one bad choice of θ for the associated interval; hence the overall probability θ is bad is at most $K|D|/b$. Thus it suffices to show that whenever θ is not bad, C is satisfied by α.

For each $t \in [k]$ let $B_t = \mathrm{supp}_\theta(p_{S_t})$; these sets are nonempty because $\theta \leq r^{-1} \leq |D|^{-1}$. Also, let $\mathcal{J} = \{J \in B_1 \times \cdots \times B_k : R(j) = 1\}$. By Lemma 2.2, to show that C is satisfied by α, we only need to show that $J \subseteq_S B_1 \times \cdots \times B_k$ — i.e., that for all $t \in [k]$ and all $j \in B_t$ there exists a tuple $J \in \mathcal{J}$ such that $J_t = j$. We show this is true for $t = k$ and the statement for other values of t follows in the same way. For any $j \in B_k$, we have $\theta \leq p_{S_k}(j)$ by the definition of B_t. Since θ is not bad, we know that $\theta \notin (p_{S_k}(j)/r, p_{S_k}(j)]$. Therefore we have $\theta \leq p_{S_k}(j)/r$. Now since all but at most $\epsilon_i \leq \sqrt{\epsilon}$ of the probability mass in q_C is on assignments satisfying R, we conclude

$$\sum_{J' \in D^{k-1} : R(J', j) = 1} q_C(J', j) \geq p_{S_k}(j) - \sqrt{\epsilon} \geq p_{S_k}(j)/2.$$

Here we used $2\sqrt{\epsilon} \leq r^{-b} \leq \theta \leq p_{S_k}(j)$. Now the pigeonhole principle implies there exists some $J' \in D^{k-1}$ with $R(J, j) = 1$ and $q_C(J, j) \geq p_{S_k}(j)/(2|D|^{k-1}) \geq p_{S_k}(j)/r$. By consistency of marginals this certainly implies $p_{S_{t'}}(J_{t'}) \geq p_{S_t}(j)/r \geq \theta$ for all $t' \in [k-1]$. Now for all $t' \in [k-1]$ we know that $J'_{S_{t'}} \in B_{t'}$. Therefore, if we let $J = (J', j)$ we have that $J \in \mathcal{J}$ and $J_k = j$. □

4 LP-decidable implies width 1

In this section we show the "only if" part of Theorem 1.1: i.e., that if the basic LP relaxation robustly decides CSP(Γ) then CSP(Γ) must have width 1. Now if the basic LP relaxation robustly decides CSP(Γ) then in particular it decides CSP(Γ); thus it suffices to prove the following stronger theorem:

Theorem 4.1. *Suppose the basic linear program decides* CSP(Γ)*, meaning that* $\mathrm{LPVal}(\mathcal{I}) = 1$ *implies* $\mathrm{Opt}(\mathcal{I}) = 1$ *for all instances* $\mathcal{I}$*. Then* CSP(Γ) *has width* 1.

Although the hypothesis of Theorem 4.1 refers to all possible instances $\mathcal{I}$ of CSP(Γ), it suffices to focus on a certain "most general" instance we'll call $\mathcal{M}(\Gamma)$. Let us make some definitions.

Definition 4.2. Given a finite set A, let $\triangle(A)$ denote the set of all probability measures on A. For $\ell \in \mathbb{N}^+$, let $\triangle_\ell(A) \subset \triangle(A)$ consist of those measures ν such that $\mathbf{Pr}_\nu[a] \in \ell^{-1}\mathbb{Z}$ for all $a \in A$.

Definition 4.3. Given the set of relations Γ over domain D, for any $\ell \in \mathbb{N}^+$ we define $\mathcal{M}_\ell(\Gamma)$ to be the following instance of CSP(Γ): The set of variables is $\triangle_\ell(D)$. For each k-ary relation R in Γ we impose constraint R on scope S if and only if there exists a probability measure $q_S \in \triangle_{D^k}$ supported on the satisfying assignments for R and with t'th marginal equal to S_t for all $t \in [k]$.

Definition 4.4. We similarly define the infinite "instance" $\mathcal{M}(\Gamma)$ with variable set $\triangle(D)$.[3]

Note that for any finite $\ell \in \mathbb{N}^+$ we have $\text{LPVal}(\mathcal{M}_\ell(\Gamma)) = 1$ "by construction"; in particular, there is an LP solution of value 1 in which the "p_v" distribution for variable v is simply v itself. (One may also reasonably say that $\text{LPVal}(\mathcal{M}(\Gamma)) = 1$, though we will not "use" this.) Thus if the basic LP decides CSP(Γ) it follows that $\text{Opt}(\mathcal{M}_\ell(\Gamma)) = 1$ for all ℓ. From this one can easily deduce (see Appendix C):

Proposition 4.5. *Suppose the basic LP decides* CSP(Γ) *and hence* $\mathcal{M}_\ell(\Gamma)$ *is satisfiable for each* $\ell \in \mathbb{N}^+$. *Then: 1.* $\mathcal{M}(\Gamma)$ *is also satisfiable. 2.* CSP(Γ) *has symmetric polymorphisms of every arity.*

We give a satisfying assignment for $\mathcal{M}(\Gamma)$ a special name:

Definition 4.6. If $\beta : \triangle(D) \to D$ satisfies all constraints in $\mathcal{M}(\Gamma)$ we call it a *measure operation*.

It's easy to show that if CSP(Γ) is width-1 and thus has a *set* operation g, then the assignment $\alpha(\nu) = g(\text{supp}(\nu))$ is a measure operation. To complete the proof of Theorem 4.1 we will show the converse; that CSP(Γ) having a measure operation implies it has width 1. This is the content of the following theorem:

Theorem 4.7. *Suppose* CSP(Γ) *has a measure operation. Then it has tree duality.*

Proof. Let $\beta : \triangle(D) \to D$ be a measure operation for CSP(Γ). To show tree duality we need to show that for any unsatisfiable instance $\mathcal{I}$ of CSP(Γ), there is an unsatisfiable tree instance $\mathcal{T}$ which witnesses this (i.e., is homomorphic to $\mathcal{I}$). Assuming as we may that $\mathcal{I}$ is connected, we will choose $\mathcal{T}$ to be the *universal cover tree* of $\mathcal{I}$, defined below. Although this is an infinite tree, we may obtain from it a finite witness: if $\mathcal{T}$ is unsatisfiable it will have some finite unsatisfiable subtree.

Let's now define the universal cover tree $\mathcal{T}$ of $\mathcal{I}$. (This notion originates in algebraic topology but we can get away with elementary definitions, not mentioning fundamental groups at all.) Pick an arbitrary variable x_0 in $\mathcal{I}$ to be the *base point* (different choices will lead to isomorphic instances). The variables of $\mathcal{T}$ are in one-to-one correspondence with all non-backtracking walks in $\mathcal{I}$ starting at x_0. For each relation $R \in \Gamma$ of arity k, we include an R-constraint on variables (walks) $\tau_1, \dots, \tau_k$ if and only if there exists $j \in [k]$ such that:

1. for each $i \neq j$, τ_i is an extension of τ_j by one step;
2. all walks τ_i other than τ_j have the same "last" constraint $C = (R, S)$, using relation R;
3. for each $i \neq j$, the last step in τ_i — call it $(x^i, C = (R, S), t^i, u^i, y^i)$ — has $t^i = j$ and $u^i = i$. In other words, $S = (y^1, \dots, y^{j-1}, z, y^{i+1}, \dots, y^k)$, where z is the endpoint of τ_j.

[3] Here we are stretching the definition of "instance" to allow for infinitely many variables and constraints. In this case it does not make sense to speak about "fractions of the constraints in $\mathcal{M}(\Gamma)$" but it still makes sense to ask whether $\mathcal{M}(\Gamma)$ is satisfiable.

The universal cover tree is indeed a "tree" with a natural homomorphism to $\mathcal{I}$, mapping a non-backtracking walk to its endpoint. Given any two walks σ and τ in $\mathcal{I}$ for which the endpoint of σ is the starting point of τ, we can define their concatenation $\sigma \circ \tau$. After the obvious cancelations we can make this concatenation non-backtracking in a unique way. We will actually only need this definition for σ a non-backtracking walk from x_0 to x_0; in this case we can define the iterated concatenation $\alpha^c = \alpha \circ \alpha \circ \cdots \circ \alpha$ (c times).

Recall our goal is to show that if $\mathcal{I}$ is unsatisfiable then so too is $\mathcal{T}$. We will show the contrapositive. So assume $\mathcal{T}$ is satisfiable; we need to prove that $\mathcal{I}$ is also satisfiable. Let $\sigma_1, \sigma_2, \sigma_3, \ldots$ be an enumeration of all non-backtracking walks in $\mathcal{I}$ which start and end at the base point x_0. Our goal will be to show that there is a satisfying assignment α for $\mathcal{T}$ which is *σ_i-periodic* for all i. By σ_i-periodic we mean that $\alpha(\tau) = \alpha(\sigma_i \circ \tau)$ for all variables τ in $\mathcal{T}$. Such an α immediately yields a satisfying assignment for $\mathcal{I}$.

We inductively construct α by constructing satisfying assignments $\alpha_1, \alpha_2, \ldots$ for $\mathcal{T}$, where α_j is σ_i-periodic for all $i < j$. The existence of α_1 follows from the assumption that $\mathcal{T}$ is satisfiable. We now show how to construct α_{j+1} from α_j. It suffices to construct an LP solution p with $\mathrm{LPVal}_{\mathcal{T}}(p) = 1$ which is σ_i-periodic for all $i < j+1$; we can then compose this with the measure operation β to produce the required satisfying assignment α_{j+1}.

For $\ell \in \mathbb{N}^+$, define p^ℓ be the LP solution for $\mathcal{T}$ in which p^ℓ_τ is the probability distribution on D given by outputting $\alpha_j(\sigma_j^c \circ \tau)$ for a uniformly random $c \in [\ell]$. Each p^ℓ in fact has LP-value 1, being a convex combination of satisfying assignments (note that if $R(\tau_1, \ldots, \tau_k) = 1$ is a constraint in $\mathcal{T}$ then so is $R(\sigma_j^c \circ \tau_1, \ldots, \sigma_j^c \circ \tau_k) = 1$ for any c). Since $\triangle(D)$ is compact and $\mathcal{T}$ is countable we can pass to a subsequence $(p^{\ell_n})_n$ of $(p^\ell)_\ell$ for which $p^{\ell_n}_\tau$ is convergent for all variables τ. We may now define p to be the LP solution in which $p_\tau = \lim_{n\to\infty} p^{\ell_n}_\tau$. It is not hard to verify that $\mathrm{LPVal}_{\mathcal{T}}(p) = 1$ since $\mathrm{LPVal}_{\mathcal{T}}(p^{\ell_n}) = 1$ for all n. The LP solution p is also σ_i-periodic for all $i < j$ since each p_ℓ is. Finally, note that for every ℓ the statistical difference of p^ℓ_τ and $p^\ell_{\alpha_j \circ \tau}$ is at most $2/\ell$; it follows that $p_\tau = p_{\alpha_j \circ \tau}$ for all variables τ; i.e., p is α_j-periodic as required.

Having produced the sequence of satisfying assignments $\alpha_1, \alpha_2, \ldots$ for $\mathcal{T}$ in which α_j is σ_i-periodic for all $i < j$, it remains to produce a satisfying assignment α for $\mathcal{T}$ which is σ_i-periodic for all i. We do this by passing to a subsequence $(\alpha_{j_n})_n$ such that for each variable τ in $\mathcal{T}$ the sequence $(\alpha_{j_n}(\tau))_n$ is eventually constant. We define $\alpha(\tau)$ to be this eventually constant value. The assignment α indeed completely satisfies $\mathcal{T}$ because for each constraint $C = (R, S)$ in $\mathcal{T}$, the subsequence of satisfying assignments $(\alpha_{j_n})_n$ eventually becomes unchanging when restricted to S. Finally, for any i the assignment α is σ_i-periodic because the subsequence $(\alpha_{j_n})_n$ only fails to be σ_i-periodic for finitely many n. The proof is complete. □

5 Conclusions

We have shown that the basic linear programming relaxation robustly decides a CSP if and only if the CSP has width 1. We view this as a first step towards a proof of the Guruswami–Zhou conjecture, that a CSP is efficiently robustly decidable if and only if it has bounded width (assuming $\mathsf{P} \neq \mathsf{NP}$).

References

[1] Libor Barto and Marcin Kozik. Constraint satisfaction problems of bounded width. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, volume 9, pages 595–603. 1, 2.3

[2] Andrei Bulatov. Combinatorial problems raised from 2-semilattices. *Journal of Algebra*, 298(2):321–339, 2006. D.2, D.3

[3] Andrei Bulatov. Bounded relational width. Available at http://www.cs.sfu.ca/∼abulatov/mpapers.html, 2009. 1, 2.3

[4] Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005. 1

[5] Catarina Carvalho, Víctor Dalmau, and Andrei Krokhin. Two new homomorphism dualities and lattice operations. *Journal of Logic and Computation*, 2010. A.1

[6] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for Unique Games. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 205–214, 2006. 1

[7] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Note on MAX 2SAT. In *Electronic Colloquium on Computational Complexity TR06-064*, 2006. 1

[8] Víctor Dalmau and Justin Pearson. Closure functions and width 1 problems. In *Proceedings of the 5th Annual Principles and Practice of Constraint Programming*, pages 159–173, 1999. 1.1, 2.3

[9] Tomás Feder and Moshe Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998. 1, 1.1, 2.1, 2.1, 2.3

[10] Michel Goemans and David Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995. 1

[11] Venkatesan Guruswami and Yuan Zhou. Tight bounds on the approximability of almost-satisfiable Horn SAT and Exact Hitting Set. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1574–1589, 2011. 1, 1, 1.1, A

[12] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. 1

[13] Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997. 1, 2.3

[14] David Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974. 1

[15] Peter Jonsson, Andrei Krokhin, and Fredrik Kuivinen. Hard constraint satisfaction problems have hard gaps at location 1. *Theoretical Computer Science*, 410(38–40):3856–3874, 2009. 1

[16] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2000. A

[17] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th ACM Symposium on Theory of Computing*, pages 767–775, 2002. 1, 1

[18] Gábor Kun and Mario Szegedy. A new line of attack on the Dichotomy Conjecture. In *Electronic Colloquium on Computational Complexity TR09-059*, 2009. 1, 2

[19] Alan Mackworth. Consistency in networks of relations. *Artificial intelligence*, 8(1):99–118, 1977. 1.1

[20] Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 245–254, 2008. 1, 1.1

[21] Yuichi Yoshida. Optimal constant-time approximation algorithms and (unconditional) inapproximability results for every bounded-degree CSP. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, pages 665–674, 2011. B.3

[22] Uri Zwick. Finding almost-satisfying assignments. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 551–560, 1998. 1, 1.1

A Lattice CSPs: better quantitative dependence on ϵ

As discussed at the end of Section 1.1, one cannot hope to improve the approximation guarantee of $1 - O(1/\log(1/\epsilon))$ given by our LP-rounding algorithm, even in the case of Horn-3Sat. On the other hand, for Horn-2Sat it is known [16] that on $(1-\epsilon)$-satisfiable instances one can efficiently find $(1 - O(\epsilon))$-satisfying assignments (indeed, $(1 - 2\epsilon)$-satisfying [11]). One might ask what the algebraic difference is between Horn-2Sat and Horn-3Sat. A notable difference is that the former is a *lattice* CSP.

Subclasses of width-1: lattice and semilattice CSPs. A broad natural subclass of the width-1 CSPs is the class of *semilattice* CSPs. These are CSPs which have a *semilattice polymorphism*, meaning a binary polymorphism $\wedge$ which is associative, commutative, and idempotent. Horn-Sat CSPs are not just width-1 but are in fact semilattice; thus we cannot hope for improved dependence on ϵ even for semilattice CSPs.[4]

An even further subclass is that of *lattice* CSPs. These are CSPs whose relations are preserved by *two* semilattice operations $\wedge$ and $\vee$ which additionally satisfy the "absorption" identity: $\vee(x, \wedge(x, y)) = \wedge(x, \vee(x, y)) = x$. Note that $\vee$ and $\wedge$ extend naturally to polymorphisms of every arity. Good examples of lattice CSPs are "lattice retraction problems". Here there is a fixed lattice poset L; the CSP's domain is L and its constraints are the poset constraint "$\leq$" along with all unary constraints "$=_a$" for $a \in L$.

A.1 Robust decidability for lattice CSPs

In this section we prove a variant of our Theorem 3.1 which shows an efficient LP-based algorithm for finding $(1 - O(\epsilon))$-satisfying assignments to $(1-\epsilon)$-satisfiable *lattice* CSP instances.

We first describe the characterization of lattice CSPs we need. Carvalho, Dalmau, and Krokhin [5] have observed that if CSP(Γ) has lattice polymorphisms then it is preserved by what they call an *absorptive block-symmetric operation*. This is an operation f which takes as input tuples (of any

[4] There are CSPs which are width-1 but not semilattice; e.g., the CSP over domain $\{a, b, c, d\}$ with all unary relations and also the binary relations (a, b), (b, a), (c, a), (c, b), (c, d), (d, c), and (d, d).

positive length) of nonempty subsets of D, outputs an element of D, and has the following properties:

- (Block-symmetry.) $f(B_1, \ldots, B_\ell)$ only depends on $\{B_1, \ldots, B_\ell\}$.
- (Absorption.) If $B \supseteq B_1$ then $f(B, B_1, \ldots, B_\ell) = f(B_1, \ldots, B_\ell)$.
- (Preservation.) Let R be an arity-k relation Γ and let $(B_i^j)_{i=1\ldots\ell}^{j=1\ldots k}$ be nonempty subsets of D. Assume that for each $i \in [\ell]$ there is a $\mathcal{J}_i \subseteq_R B_i^1 \times \cdots \times B_i^k$ all of whose members satisfy R. Then $R(f(B_1^1, \ldots, B_\ell^1), \ldots, f(B_1^k, \ldots, B_\ell^k))) = 1$.

Indeed, the operation f is simply $f(B_1, \ldots, B_\ell) = \bigvee\{\wedge B_i : i \in [\ell]\}$.

We now show:

Theorem A.1. *Let Γ be a finite set of relations over the finite domain D, each relation having arity at most K. Assume that* CSP(Γ) *has lattice polymorphisms. Then there is a* poly(n, m)*-time algorithm for* CSP(Γ) *which when given an input $\mathcal{I}$ with* $\text{LPVal}(\mathcal{I}) = 1 - \epsilon$ *outputs an assignment $\alpha : V \to D$ with* $\text{Val}_{\mathcal{I}}(\alpha) \geq 1 - O(K2^{|D|})\epsilon$.

Proof. As in Theorem 3.1, the first step of the algorithm is to solve the LP relaxation of the instance, determining an optimal solution $\{p_v : v \in V\}$, $\{q_{C_i} : i \in [m]\}$ which obtains $\text{LPVal}(\mathcal{I}) = 1 - \epsilon$. Since CSP($\Gamma$) has lattice polymorphisms, it has some absorptive block-symmetric operation f. We next describe a randomized LP-rounding algorithm:

1. Set $r = (2K2^{|D|}m)^{-1}$ and choose $\theta \in \{1/r, 2/r, 3/r, \ldots, 1\}$ uniformly at random.
2. For each $v \in V$, define $\mathcal{B}_v = \{B \subseteq D : p_v(B) \geq \theta\}$, a nonempty family of nonempty sets. (Here we introduce the notation $p_v(B) = \sum_{b \in B} p_v(b)$.)
3. Output the assignment $\alpha : V \to D$ defined by $\alpha(v) = f(\mathcal{B}_v)$.

We will show for each constraint $C_i = (S_i, R_i)$ that

$$\mathbf{Pr}[R_i(\alpha(S_i)) = 0] \leq K2^{|D|}(\epsilon_i + 1/r) = K2^{|D|}\epsilon_i + 1/2m, \tag{5}$$

where $\epsilon_i = \sum_{J : R_i(J) = 0} q_{C_i}(J)$ as in the previous proof. It then follows from linearity of expectation that the expected fraction of constraints not satisfied by α is at most $K2^{|D|}\,\text{avg}\{\epsilon_i\} + 1/2m = K2^{|D|}\epsilon + 1/2m$. We can therefore efficiently deterministically find an α with value at least $1 - K2^{|D|}\epsilon - 1/2m$ by trying all $O(m)$ possible values for θ. This is sufficient to prove the theorem: if $\epsilon < (2K2^{|D|}m)^{-1}$ then α's value exceeds $1 - 1/m$ and hence is in fact 1; if $\epsilon \geq (2K2^{|D|}m)^{-1}$ then the $O(\cdot)$ in the theorem statement covers the loss of $1/2m$.

We now give the analysis justifying (5) for each fixed constraint $C_i = (S_i, R_i)$. For simplicity we henceforth write $C = C_i$, $S = S_i$, $R = R_i$ and suppose that R has arity $k \leq K$. It suffices to show that $R(\alpha(S)) = 1$ holds assuming

$$\theta \notin (p_{S_t}(B), p_{S_t}(B) + \epsilon_i] \quad \forall t \in [k],\ \forall B \subseteq D. \tag{6}$$

The reason is that the probability of (6) not holding is at most $K2^{|D|}(\epsilon_i + 1/r)$. Note that with assumption (6), whenever we have $p_{S_t}(B) \geq \theta - \epsilon_i$ it follows that in fact $p_{S_t}(B) \geq \theta$ and thus $B \in \mathcal{B}_{S_t}$.

Claim A.2. *For all $t \in [k]$ and $B \in \mathcal{B}_{S_t}$, there exist $B_1, \ldots, B_k$ with $B_u \in \mathcal{B}_{S_u}$ such that: a) $B_t \subseteq B$; b) there exists $\mathcal{J} \subseteq_S B_1 \times \cdots \times B_k$ with $R(J) = 1$ for all $J \in \mathcal{J}$.*

Proof. Suppose $B \in \mathcal{B}_{S_t}$, so $p_{S_t}(B) \geq \theta$. Letting $\mathcal{J}' = \{J \in D^k : J_t \in B\}$, it follows from consistency of marginals that $q_C(\mathcal{J}') \geq \theta$. Thus if $\mathcal{J}$ is the subset of $\mathcal{J}'$ for which R holds, it follows that $q_C(\mathcal{J}) \geq \theta - \epsilon_i$. For $u \in [k]$, we define $B_u = \{J_u : J \in \mathcal{J}\}$. Certainly $B_t \subseteq B$, and by consistency of marginals we obtain from $q_C(\mathcal{J}) \geq \theta - \epsilon_i$ that $p_{S_u}(B_u) \geq \theta - \epsilon_i$ for each $u \in [k]$. Thus it follows from assumption (6) that $B_u \in \mathcal{B}_{S_u}$ for each u, completing the proof of the claim. □

For each choice of $t \in [k]$ and $B \in \mathcal{B}_{S_t}$, take the (names of the) sets $B_1, \ldots, B_k$ given by the above claim and arrange them in a height-k column. Adjoin all of these columns to form a $k \times \ell$ matrix M, where $\ell = \sum_{t=1}^{k} |\mathcal{B}_{S_t}|$. The matrix M has the following properties: (i) each entry in row u is a set in $\mathcal{B}_{S_u}$; (ii) for each set $B \in \mathcal{B}_{S_u}$, some subset of it appears in the u^{th} row of M; (iii) for each column $(B_1, \ldots, B_k)$ of M there is a $\mathcal{J} \subseteq_S B_1 \times \cdots \times B_k$ all of whose members satisfy R.

Suppose we now apply the absorptive block-symmetric operation f to the rows of M, with the u^{th} row producing $j_u \in D$. By (iii), $R(j_1, \ldots, j_k) = 1$. Thus the justification of (5) is complete if we can show $j_u = f(\mathcal{B}_{S_u}) = \alpha(S_u)$. But this follows from (i), (ii), and the absorptive property of f. □

B Connection to Property Testing

In property testing, given an instance as an oracle access, we want to test the instance satisfies some predetermined property or ϵ-far from satisfying the property. Here, the definition of ϵ-farness depends on each model. We want to design algorithms that run in constant time, i.e., which depends only on ϵ and not on sizes of inputs. In this section, we show that, in the model so-called bounded-degree model, the satisfiability of an instance in CSP(Γ) is testable in constant time if and only if CSP(Γ) has width 1.

First, we define the *bounded-degree model*. Let $\mathcal{I}$ be an instance of CSP with a variable set V and a constraint set $\mathcal{C}$. We define the degree of a variable $v \in V$ as the number of constraints containing v. In the bounded-degree model, we only consider instances such that every variable has degree at most d where d is a predetermined constant. An instance $\mathcal{I}$ is called *ϵ-far* from satisfiability if we must remove at least ϵdn constraints in order to make it satisfiable. An instance is represented as an oracle $O_{\mathcal{I}} : V \times [d] \to \mathcal{C}$; If we specify a variable $v \in V$ and an index $i \in [d]$, $O_{\mathcal{I}}$ returns the i-th constraint containing v. If no such constraint exists, $O_{\mathcal{I}}$ returns a special symbol $\perp$.

Definition B.1. An algorithm is called an testing algorithm for CSP(Γ) if, given an instance $\mathcal{I}$ as an oracle access $O_{\mathcal{I}}$, it accepts with probability at least 2/3 when $\mathcal{I}$ is satisfiable, and it rejects with probability at least 2/3 when $\mathcal{I}$ is ϵ-far from satisfiability.

It is known that the testability of CSP(Γ) is described by the integrality gap curve.

Definition B.2. The integrality gap curve for a CSP(Γ) with respect to the basic LP is defined as

$$S_\Gamma(c) = \inf_{\mathcal{I} \in \Gamma, \mathrm{LP}(\mathcal{I}) \geq c} \mathrm{Opt}(\mathcal{I}).$$

Lemma B.3 (Theorem 1.4 of [21])**.** *Let D be a finite domain and Γ be a finite set of relations over D. Then, in the bounded-degree model,*

1. *if* $\mathrm{CSP}(\Gamma)$ *satisfies* $S_\Gamma(1) = 1$ *and* $S_\Gamma(c)$ *is continuous at* $c = 1$*, then there exists a testing algorithm for* $\mathrm{CSP}(\Gamma)$ *with* $O(1)$ *queries.*

2. *if* $\mathrm{CSP}(\Gamma)$ *satisfies* $S_\Gamma(1) < 1$*, then any testing algorithm for the* $\mathrm{CSP}(\Gamma)$ *requires* $\Omega(\sqrt{n})$ *queries.*

Note that the notation $O(\cdot)$ hides constants such as $\epsilon, d, |D|$ and $|\Gamma|$.

Theorem B.4. *Let* D *be a finite domain and* Γ *be a finite set of relations over* D*. Then, in the bounded-degree model,*

1. *if* $\mathrm{CSP}(\Gamma)$ *has width* 1*, then there exists a testing algorithm for* $\mathrm{CSP}(\Gamma)$ *with* $O(1)$ *queries.*

2. *if* $\mathrm{CSP}(\Gamma)$ *does not have width* 1*, then testing* $\mathrm{CSP}(\Gamma)$ *requires* $\Omega(\sqrt{n})$ *queries.*

Proof. We first show the former part. Let $\mathcal{I}$ be an instance of $\mathrm{CSP}(\Gamma)$. Form Theorem 3.1 $\mathrm{LP}(\mathcal{I}) = 1$ implies $\mathrm{Opt}(\mathcal{I}) = 1$. Thus, $S_\Gamma(1) = 1$. Also, from Theorem 3.1 $\mathrm{LP}(\mathcal{I}) = 1 - \epsilon$ implies $\mathrm{Opt}(\mathcal{I}) \geq 1 - 1/\log(1/\epsilon)$. It follows that $\lim_{c \to 1} S_\Gamma(c) = 1$, which means that $S_\Gamma(c)$ is continuous at $c = 1$. Then, the claim holds from the former part of Lemma B.3.

Now, we show the latter part. From Theorem 4.1 there exists an instance $\mathcal{I}$ of $\mathrm{CSP}(\Gamma)$ such that $\mathrm{LP}(\mathcal{I}) = 1$ and $\mathrm{Opt}(\mathcal{I}) < 1$. This indicates $S_\Gamma(1) < 1$. Thus, the claim holds from the latter part of Lemma B.3. □

C Minor proof details for Section 4

Here we prove Proposition 4.5.

Proof. As we described, by construction $\mathrm{LPVal}(\mathcal{M}_\ell(\Gamma)) = 1$ for each $\ell \in \mathbb{N}^+$. Since the basic LP decides $\mathrm{CSP}(\Gamma)$ it follows that there is in fact a satisfying assignment α_ℓ for $\mathcal{M}_\ell(\Gamma)$. We use this to show the two statements.

First we need to show that $\mathcal{M}(\Gamma)$ is satisfiable. It suffices to show that for every finite subset V of variables in $\mathcal{M}(\Gamma)$ the induced sub-instance is satisfiable; the result then follows from compactness. Suppose first that all variables in V are "rational" probability measures, meaning that they give rational probabilities to all elements of D. Then $V \subseteq \triangle_{\ell^*}(D)$ for some sufficiently large ℓ^* and it follows that the sub-instance induced by V is satisfiable, since $\mathcal{M}_{\ell^*}(\Gamma)$ is satisfiable. In general, the variables in V need not be rational. However we claim that the sub-instance they induce is isomorphic to a (not necessarily induced) sub-instance of $\mathcal{M}(\Gamma)$ with rational probability measures, and thus is satisfiable as before. The justification for the claim is that the true statement "the sub-instance induced by V is isomorphic to a sub-instance of $\mathcal{M}(\Gamma)$" is expressible by a linear program with integer coefficients; thus the linear program has a feasible *rational* solution. This completes the proof that the sub-instance induced by V is satisfiable, and thus $\mathcal{M}(\Gamma)$ is satisfiable.

We now moving on to the second statement of the proposition, showing that $\mathrm{CSP}(\Gamma)$ has a symmetric polymorphism of arity ℓ for each $\ell \in \mathbb{N}^+$. Consider the operation $f : D^\ell \to D$ defined by $f(x) = \alpha_\ell(p_x)$, where p_x is defined to be the probability measure on D given by choosing a uniformly random coordinate of the ℓ-tuple x. (Note that indeed $p_x \in \triangle_\ell(D)$ so this is well-defined.) The operation f is symmetric, and it remains to check that it is a polymorphism. To see this, let R be any relation in Γ of arity, say, k, and fix $x^1, \ldots, x^k \in D^\ell$ satisfying $R(x^1_i, \ldots, x^k_i) = 1$ for all $i \in [\ell]$. Let $S \in V^k$ be the tuple with $S_t = p_{x^t}$ and let q_S be the probability measure on D^k

given by choosing $(x_i^1, \dots, x_i^k)$ for uniformly random $i \in [\ell]$. By virtue of this q_S, the constraint (R, S) appears in $\mathcal{M}_\ell(\Gamma)$. Since α_ℓ satisfies this constraint we have $R(\alpha_\ell(p_{x^1}), \dots, \alpha_\ell(p_{x^k})) = 1$; i.e., $R(f(x^1), \dots, f(x^k)) = 1$. Thus f is indeed a polymorphism. □

D Another proof of Theorem 3.1

In this section, we present another proof of Theorem 3.1. The proof is slightly lengthy but gives better dependency on K and $|D|$.

Theorem D.1. *Let Γ be a finite set of relations over the finite domain D, each relation having arity at most K. Assume that* CSP(Γ) *has width* 1. *Then there is a* poly(n, m)*-time algorithm for* CSP(Γ) *which when given an input $\mathcal{I}$ with* LPVal$(\mathcal{I}) = 1 - \epsilon$ *outputs an assignment $\alpha : V \to D$ with* $\mathrm{Val}_{\mathcal{I}}(\alpha) \geq 1 - O(K|D|\log(K|D|)/\log(1/\epsilon))$. *(In particular,* $\mathrm{Val}_{\mathcal{I}}(\alpha) = 1$ *if* LPVal$(\mathcal{I}) = 1$.*)*

The basic idea of the proof is as follows. We first solve the basic linear program and discard constraints that are not well-satisfied by the LP solution. To round the LP solutions for the remaining constraints, we simulate a propagation algorithm for CSPs of width 1, called 1-Minimality. The algorithm iteratively restricts the domain of variables using current constraints. For CSPs of width 1, if the algorithm cannot perform propagation anymore and there is no empty relation, the instance is satisfiable. In our rounding, instead, we discard constraints to stop the propagation. By carefully choosing constraints to be removed, we can obtain a large number of constraints for which 1-Minimality cannot continue propagation. Thus, the resulting instance is satisfiable, and we can compute the desired assignment from the instance. Our algorithm can be seen as a generalization of Zwick's algorithm for Horn-kSat.

In the rest of the section, we give a detailed proof of Theorem D.1.

D.1 Weighted constraint satisfaction problems

We will prove Theorem D.1 for *weighted CSPs* defined as follows:

Definition D.2. A weighted instance of the constraint satisfaction problem is a tuple $\mathcal{I} = (V, D, \mathcal{C}, \boldsymbol{w})$ where V is a finite set of variables, D is a finite domain, $\mathcal{C}$ is a finite set of constraints and $\boldsymbol{w} : \mathcal{C} \to [0, \infty)$ is a weight function. The objective is to find an assignment $\alpha : V \to D$ that maximizes the total weight of the constraints satisfied by α.

For a CSP instance $\mathcal{I} = (V, D, \mathcal{C}, \boldsymbol{w})$ and a set of constraints $\mathcal{C}' \subseteq \mathcal{C}$, we define $\boldsymbol{w}_{\mathcal{I}} = \sum_{C \in \mathcal{C}} \boldsymbol{w}_C$ and $\boldsymbol{w}_{\mathcal{C}'} = \sum_{C \in \mathcal{C}'} \boldsymbol{w}_C$. Throughout the paper, we assume that any input instance $\mathcal{I} = (V, D, \mathcal{C}, \boldsymbol{w})$ of the CSP satisfies $\boldsymbol{w}_{\mathcal{I}} = 1$. For an assignment α, we define $\mathrm{Val}_{\mathcal{I}}(\alpha)$ as the total weight of the constraints satisfied by α. Also, we define $\mathrm{Opt}(\mathcal{I}) = \max_\alpha\{\mathrm{Val}_{\mathcal{I}}(\alpha)\}$ as the optimal value. We modify the objective function of the basic linear program as

$$\mathrm{LPVal}_{\mathcal{I}}(\{p_v\}, \{q_C\}) = \sum_{C=(S,R) \in \mathcal{C}} \boldsymbol{w}_C \Pr_{J \sim q_C}[R(J) = 1].$$

D.2 (k, ℓ)-Minimality

We need a different characterization of width based on a simple propagation algorithm, called (k, ℓ)-Minimality [2] (see Algorithm 1). When $k = \ell$, it is also called k-Minimality. We need the following definition to describe the behavior of (k, ℓ)-Minimality.

Algorithm 1 (k,ℓ)-Minimality

Input: $\mathcal{I} = (V, D, \mathcal{C}_\mathcal{I})$.
$\mathcal{I}^0 := (V, D, \mathcal{C}^0_\mathcal{I} \cup \mathcal{C}^0_\mathcal{D})$ where $\mathcal{C}^0_\mathcal{I} = \mathcal{C}_\mathcal{I}$ and $\mathcal{C}^0_\mathcal{D} = \{(S, D^\ell) \mid S \subseteq V, |S| = \ell\}$.
for $i = 1$ to ∞ **do**
  $\mathcal{J}^{i-1} := \mathcal{I}^{i-1}$.
  for each $W \subseteq V, |W| = k$ **do**
    Let $\mathcal{S}^{i-1}_W$ be the set of all satisfying assignments to $\mathcal{I}^{i-1}_W$
    for each constraint $C = (S, R)$ of $\mathcal{J}^{i-1}$ **do**
      replace C with (S, R') where $R' = \{J \in R \mid \mathrm{pr}_{S\cap W} J \in \mathrm{pr}_{S\cap W} \mathcal{S}^{i-1}_W\}$.
  $\mathcal{I}^i = (V, D, \mathcal{C}^i_\mathcal{I} \cup \mathcal{C}^i_\mathcal{D}) := \mathcal{J}^{i-1}$
  if $\mathcal{I}^i = \mathcal{I}^{i-1}$ **then**
    return $\mathcal{I}^i$

For a tuple $J = (J_1, \ldots, J_n)$ and a subset $W = \{W_1, \ldots, W_k\} \subseteq [n]$, we define the *projection* of J to W by $\mathrm{pr}_W J = (J_{W_1}, \ldots, J_{W_k})$. The projection of a relation $R \subseteq D^n$ to W is defined by $\mathrm{pr}_W R = \{\mathrm{pr}_W J \mid J \in R\}$. For a CSP instance $\mathcal{I} = (V, D, \mathcal{C})$ and a subset $W \subseteq V$, we define a *partial instance* $\mathcal{I}_W = (W, D, \mathcal{C}_W)$ where $\mathcal{C}_W = \{(S \cap W, \mathrm{pr}_{S\cap W} R) \mid (S, R) \in \mathcal{C}\}$. We denote by $\mathcal{S}_W$ the set of all satisfying assignments to $\mathcal{I}_W$. When $W = \{v\}$, we write $\mathcal{I}_v$ and $\mathcal{S}_v$ instead of $\mathcal{I}_W$ and $\mathcal{S}_W$, respectively.

Definition D.3 ([2])**.** Let k, ℓ be integers with $0 < k \leq \ell$. An instance $\mathcal{I} = (V, D, \mathcal{C})$ is called (k,ℓ)-*minimal* if

1. for every $W \subseteq V, |W| \leq \ell$, there exists some constraint $(S, R) \in \mathcal{C}$ such that $W \subseteq S$.
2. for every $W \subseteq V, |W| \leq k$ and two constraints $(S_1, R_1), (S_2, R_2) \in \mathcal{C}$, we have $\mathrm{pr}_{S_1\cap W} R_1 = \mathrm{pr}_{S_2\cap W} R_2$. That is, $\mathcal{S}_{S\cap W} = \mathrm{pr}_{S\cap W} R$ holds for every $(S, R) \in \mathcal{C}$ and $W \subseteq V, |W| \leq k$.

A (k,k)-minimal instance is also called k-*minimal*.

(k,ℓ)-Minimality first adds constraints of the form (S, D^ℓ) for every $S \subseteq V, |S| = \ell$ to satisfy the first condition of the (k,ℓ)-minimality. Then, it iteratively removes tuples from relations in each constraint so that the modified constraints satisfy the second condition of the (k,ℓ)-minimality. We can see that it transforms an instance to a (k,ℓ)-minimal instance with the same set of satisfying assignments. It is sound in the sense that if it outputs an instance with an empty constraint, then the instance is unsatisfiable. However, it is not complete in the sense that even if it outputs an instance with no empty constraint, the instance may be unsatisfiable. We say that a CSP has width (k,ℓ) if it is also complete. A formal definition is given below.

Definition D.4. Let Γ be a constraint language. We say that CSP(Γ) has width (k,ℓ) if the following holds: if (k,ℓ)-Minimality on an instance $\mathcal{I}$ of CSP(Γ) outputs an instance $\mathcal{I}'$ with no empty constraint, then $\mathcal{I}$ is always satisfiable. Moreover, CSP(Γ) has width k if Γ has width (k,k). Also, CSP(Γ) has bounded width if Γ has width k for some constant k.

Let us see several properties of the algorithm 1-Minimality. Let $\mathcal{C}^i = \mathcal{C}^i_\mathcal{I} \cup \mathcal{C}^i_\mathcal{D}$ be the set of constraints in $\mathcal{I}^i$ where $\mathcal{C}^i_\mathcal{I}$ (resp., $\mathcal{C}^i_\mathcal{D}$) is the set of constraints originated from $\mathcal{C}_\mathcal{I}$ (resp., $\mathcal{C}_\mathcal{D}$). For a variable $v \in V$, we define $\mathcal{S}^i_v$ as the set of all satisfying assignments to $\mathcal{I}^i_v$. For convenience, we define $\mathcal{S}^{-1}_v = D$.

Lemma D.5. *Let $i \geq 0$ and $C = (\{v\}, R)$ be a constraint in $\mathcal{C}^i_\mathcal{D}$. Then, $R = \mathcal{S}^{i-1}_v$.*

Proof. We prove the claim by induction on i. When $i = 0$, the claim is obvious since $R = D = \mathcal{S}_v^{-1}$.

Suppose that the claim holds for $i-1$, and let $C' = (\{v\}, R') \in \mathcal{C}_{\mathcal{D}}^{i-1}$ be the constraint in $\mathcal{I}^{i-1}$ corresponding to C. From the hypothesis, $R' = \mathcal{S}_v^{i-2}$ holds. Then, we have $R = \{j \in R' \mid j \in \mathcal{S}_v^{i-1}\} = \mathcal{S}_v^{i-1}$ since $\mathcal{S}_v^{i-1} \subseteq \mathcal{S}_v^{i-2}$. □

Corollary D.6. *Let $\mathcal{I}' = (V, D, \mathcal{C}_{\mathcal{I}}^{i-1} \cup \mathcal{C}_{\mathcal{D}}^{i})$ and $\mathcal{S}'_v$ be the set of all satisfying assignments to $\mathcal{I}'_v$. Then, $\mathcal{S}'_v = \mathcal{S}_v^{i-1}$ holds for any variable $v \in V$.*

Proof. From Lemma D.5, there exists $C = (\{v\}, \mathcal{S}_v^{i-1}) \in \mathcal{C}_{\mathcal{D}}^{i}$. Since C exists in $\mathcal{I}'$, we have $\mathcal{S}'_v \subseteq \mathcal{S}_v^{i-1}$. We show the other direction. For a constraint $(S, R) \in \mathcal{C}_{\mathcal{I}}^{i-1}$ with $v \in S$, we have $\mathcal{S}_v^{i-1} \subseteq \mathrm{pr}_v R$. Also, the only constraint in $\mathcal{C}_{\mathcal{D}}^{i}$ containing v in its scope is $(\{v\}, \mathcal{S}_v^{i-1})$. Thus, $\mathcal{S}_v^{i-1} \subseteq \mathcal{S}'_v$. □

In the following two lemmas, we show that we can obtain 1-minimal instance by discarding constraints in the process of 1-Minimality.

Lemma D.7. *Let $i \geq 0$ and $\mathcal{C}'$ be the set of constraints in $\mathcal{C}_{\mathcal{I}}^{i}$ such that the corresponding constraints in $\mathcal{J}^i$ are updated at Line 8. Then, $\widetilde{\mathcal{I}} = (V, D, (\mathcal{C}_{\mathcal{I}}^{i} \setminus \mathcal{C}') \cup \mathcal{C}_{\mathcal{D}}^{i})$ is 1-minimal.*

Proof. We define $\widetilde{\mathcal{S}}_v$ as the set of all satisfying assignments to $\widetilde{\mathcal{I}}_v$. Suppose that $\widetilde{\mathcal{I}}$ is not 1-minimal. Then, there exists $C = (S, R) \in (\mathcal{C}_{\mathcal{I}}^{i} \setminus \mathcal{C}') \cup \mathcal{C}_{\mathcal{D}}^{i}$ such that $\widetilde{\mathcal{S}}_v \subsetneq \mathrm{pr}_v R$ for some $v \in S$. We can see that $C \in \mathcal{C}_{\mathcal{D}}^{i}$. Indeed, if $C \in \mathcal{C}_{\mathcal{I}}^{i} \setminus \mathcal{C}'$ holds, C must be contained in $\mathcal{C}'$ from the construction of $\mathcal{C}'$.

Then, we have $C = (\{v\}, R) \in \mathcal{C}_{\mathcal{D}}^{i}$ and $\widetilde{\mathcal{S}}_v \subsetneq \mathrm{pr}_v R = \mathcal{S}_v^{i-1}$ from Lemma D.5. This indicates that there exists $C' = (S', R') \in \mathcal{C}_{\mathcal{I}}^{i} \setminus \mathcal{C}'$ such that $\mathrm{pr}_v R' \neq \mathcal{S}_v^{i-1}$. However, if such C' exists, it should be contained in $\mathcal{C}'$ from the construction of $\mathcal{C}'$. □

Lemma D.8. *Let $i \geq 1$ and $\widetilde{\mathcal{I}} = (V, D, \mathcal{C}'_{\mathcal{I}} \cup \mathcal{C}_{\mathcal{D}}^{i})$ where $\mathcal{C}'_{\mathcal{I}} = \{(S, R) \in \mathcal{C}_{\mathcal{I}}^{i-1} \mid R$ contains exactly one tuple$\}$. Also, suppose that $\mathcal{I}^i$ has no empty constraint. Then, $\widetilde{\mathcal{I}}$ is a 1-minimal instance with no empty constraint.*

Proof. For each constraint $C = (S, R)$, we need to show that $\mathrm{pr}_v R = \widetilde{\mathcal{S}}_v$ holds for any $v \in S$. Since $\widetilde{\mathcal{S}}_v \subseteq \mathrm{pr}_v R$ is trivial, we show the other direction.

Suppose that $C \in \mathcal{C}'_{\mathcal{I}}$. Since $\mathcal{I}^i$ has no empty constraint, the unique tuple in R was not removed at Line 8 when constructing $\mathcal{I}^i$. Thus, $\mathrm{pr}_v R = \mathcal{S}_v^{i-1} = \mathcal{S}_v^{i}$ for any $v \in S$, and it follows that every constraint in $\mathcal{C}'_{\mathcal{I}}$ exists in $\mathcal{I}^i$. Then, we can see that $\widetilde{\mathcal{I}}$ is an instance obtained from $\mathcal{I}^i$ by discarding constraints. Hence, we have $\mathrm{pr}_v R = \mathcal{S}_v^{i} \subseteq \widetilde{\mathcal{S}}_v$.

Suppose that $C \in \mathcal{C}_{\mathcal{D}}^{i}$. From Lemma D.5, we have $R = \mathcal{S}_v^{i-1}$. Consider an instance $\mathcal{I}' = (V, D, \mathcal{C}_{\mathcal{I}}^{i-1} \cup \mathcal{C}_{\mathcal{D}}^{i})$. From Lemma D.6, we have $\mathcal{S}'_v = \mathcal{S}_v^{i-1}$. Since $\widetilde{\mathcal{I}}$ is obtained from $\mathcal{I}'$ by removing constraints, we have $\mathrm{pr}_v R = \mathcal{S}_v^{i-1} = \mathcal{S}'_v \subseteq \widetilde{\mathcal{S}}_v$ □

D.3 Proof of Theorem D.1

In this subsection, we prove Theorem D.1. Let Γ be a constraint language such that CSP(Γ) has width 1. Also, let $\mathcal{I} = (V, D, \mathcal{C}_{\mathcal{I}}, \boldsymbol{w})$ be an instance with an LP solution $p_v(j)$ for each $v \in V$, $j \in D$ and $q_C(J)$ for each constraint $C = (S, R)$ and tuple $J \in R$.

Our rounding algorithm is described in Algorithm 2. For a constraint $C = (S, R)$, we define $\rho(C) = \sum_{J : R(J) = 1} q_C(J)$, which means how much the constraint C is satisfied by the LP solution. At Line 3, Apx-1-Minimality first discards constraints $C \in \mathcal{C}_I$ such that $\rho(C)$ is small. After that, we basically follow the procedure of 1-Minimality. The difference from 1-Minimality is that we may (eventually, always) terminate at Lines 13 or 15 before the instance becomes 1-minimal. In this

Algorithm 2 Apx-1-Minimality

1: **Input:** $\mathcal{I} = (V, D, \mathcal{C}_\mathcal{I}, \boldsymbol{w})$ and the LP solution $\{p_v(j)\}, \{q_C(J)\}$ for $\mathcal{I}$.
2: Let $\epsilon = 1 - \sum_{C \in \mathcal{C}_\mathcal{I}} \boldsymbol{w}_C \rho(C)$, $\epsilon_1 = \sqrt{\epsilon}$ and $\epsilon_2 = \frac{K|D|\log(K|D|)}{\log(1/2\epsilon_1)}$.
3: $\mathcal{I}^0 := (V, D, \mathcal{C}^0_\mathcal{I} \cup \mathcal{C}^0_\mathcal{D})$ where $\mathcal{C}^0_\mathcal{I} = \{C \in \mathcal{C}_\mathcal{I} \mid \rho(C) \geq 1 - \epsilon_1\}$ and $\mathcal{C}^0_\mathcal{D} = \{(\{v\}, D) \mid v \in V\}$.
4: **for** $i = 1$ to ∞ **do**
5: $\quad \mathcal{J}^{i-1} := \mathcal{I}^{i-1}$.
6: $\quad$ **for** $v \in V$ **do**
7: $\quad\quad \mathcal{S}^{i-1}_v$ be the set of all satisfying assignments to $\mathcal{I}^{i-1}_v$.
8: $\quad\quad$ **for** each constraint $C = (S, R)$ in $\mathcal{J}^{i-1}$ s.t. $v \in S$ **do**
9: $\quad\quad\quad$ replace C with (S, R') where $R' = \{J \in R \mid \mathrm{pr}_v J \in \mathcal{S}^{i-1}_v\}$.
10: $\quad \mathcal{I}^i = (V, D, \mathcal{C}^i_\mathcal{I} \cup \mathcal{C}^i_\mathcal{D}, \boldsymbol{w}) := \mathcal{J}^{i-1}$
11: $\quad$ Let $\mathcal{D}^i_\mathcal{I} \subseteq \mathcal{C}^{i-1}_\mathcal{I}$ be such that the corresponding constraints in $\mathcal{J}^{i-1}$ were updated at Line 9.
12: $\quad$ **if** $\boldsymbol{w}_{\mathcal{D}^i_\mathcal{I}} < \epsilon_2 \boldsymbol{w}_{\mathcal{I}^0}$ **then**
13: $\quad\quad$ **return** $\widetilde{\mathcal{I}} = (V, D, (\mathcal{C}^i_\mathcal{I} \setminus \mathcal{D}^i_\mathcal{I}) \cup \mathcal{C}^i_\mathcal{D}, \boldsymbol{w})$.
14: $\quad$ **if** $\sum_{j=1}^i \boldsymbol{w}_{\mathcal{D}^j_\mathcal{I}} \geq K|D| \boldsymbol{w}_{\mathcal{I}^0}$ **then**
15: $\quad\quad$ **return** $\widetilde{\mathcal{I}} = (V, D, \mathcal{C}'_\mathcal{I} \cup \mathcal{C}^i_\mathcal{D}, \boldsymbol{w})$ where $\mathcal{C}'_\mathcal{I} = \{(S, R) \in \mathcal{C}^{i-1}_\mathcal{I} \mid R \text{ contains exactly one tuple}\}$)

case, we force the instance to be 1-minimal by discarding constraints. Then, we can prove that the original instance has a satisfying assignment.

For notational simplicity, though the LP solution $\{q_C(J)\}$ are indexed by constraints in $\mathcal{I}$, we regard that they are also indexed by constraints in $\mathcal{I}^i$. The following holds from Markov's inequality.

Proposition D.9. *We have* $\boldsymbol{w}_{\mathcal{I}^0} \geq 1 - \epsilon_1$. □

Lemma D.10. *Apx-1-Minimality terminates after at most* $\frac{K|D|}{\epsilon_2}$ *iterations.*

Proof. Suppose that Apx-1-Minimality never reaches Line 13. Then, $\boldsymbol{w}_{\mathcal{D}^i_\mathcal{I}} \geq \epsilon_2 \boldsymbol{w}_{\mathcal{I}^0}$ for every i. However, after $\frac{K|D|}{\epsilon_2}$ steps, $\sum_{j=1}^i \boldsymbol{w}_{\mathcal{D}^j_\mathcal{I}}$ becomes at least $K|D|$, and it must terminate at Line 15. □

We define a sequence $c_i = ((K|D|)^{i+1} - 1)/(K|D| - 1)$. Note that $c_0 = 1$ and $c_{i+1} = K|D|c_i + 1$.

Lemma D.11. *For every* $i \geq 0$ *and a constraint* $C = (S, R) \in \mathcal{C}^i_\mathcal{I}$ *in* $\mathcal{I}^i$*, we have* $\rho(C) \geq 1 - c_i\epsilon_1$.

Proof. We prove the lemma by induction on i. The case $i = 0$ is trivial since we discard all constraints violating the condition at Line 3.

Suppose that the lemma holds for $i - 1$. Now, fix a constraint $C = (S, R) \in \mathcal{C}^{i-1}_\mathcal{I}$ and consider the i-th iteration. The process of removing tuples from C can be regarded as follows. Fix a variable $v \in S$ and a value $j \in D$. if $j \notin \mathcal{S}^{i-1}_v$, then we remove from C tuples $J \in R$ such that $\mathrm{pr}_v J = j$. For each such j, there exists some constraint $C' = (S', R') \in \mathcal{C}^{i-1}$ such that $v \in S'$ and $j \notin \mathrm{pr}_v R'$. From the hypothesis, we have $p_v(j) \leq 1 - \rho(C') \leq c_{i-1}\epsilon_1$. Thus, by removing such tuples J, $\rho(C)$ decreases at most by $c_{i-1}\epsilon_1$. Since we have at most $k|D|$ choices for v and j, $\rho(C)$ decreases at most by $K|D|c_{i-1}\epsilon_1$ in total. Thus, in $\mathcal{I}^i$, $\rho(C)$ is at least $1 - c_{i-1}\epsilon_1 - K|D|c_{i-1}\epsilon_1 = 1 - c_i\epsilon_1$. □

Lemma D.12. *For every* $i \leq \frac{K|D|}{\epsilon_2} + 1$, $\mathcal{I}^i$ *has no empty constraint.*

Proof. Since $1 - c_i\epsilon_1 > 0$ for $i \leq \frac{K|D|}{\epsilon_2} + 1$ from Lemma D.11, we have $\rho(C) > 0$ for every constraint $C \in \mathcal{C}^i_\mathcal{I}$. In particular, there is no empty constraint. □

Lemma D.13. *Suppose that* Apx-1-Minimality *returns an instance* $\widetilde{\mathcal{I}}$ *at Line 13. Then,* $\widetilde{\mathcal{I}}$ *is a* 1*-minimal instance with no empty constraint and* $\boldsymbol{w}_{\widetilde{\mathcal{I}}} \geq 1 - \epsilon_1 - \epsilon_2$.

Proof. Suppose that we reach Line 13 in the t-th step. From Lemma D.7, $\widetilde{\mathcal{I}}$ is 1-minimal. From Lemma D.12, $\mathcal{I}^t$ has no empty constraint, and it follows that $\widetilde{\mathcal{I}}$ also has no empty constraint. From Proposition D.9 and $\boldsymbol{w}_{\mathcal{D}^t_{\mathcal{I}}} < \epsilon_2 \boldsymbol{w}_{\mathcal{I}^0}$, we have $\boldsymbol{w}_{\widetilde{\mathcal{I}}} \geq (1-\epsilon_2)\boldsymbol{w}_{\mathcal{I}^0} \geq 1 - \epsilon_1 - \epsilon_2$. □

Lemma D.14. *Suppose that* Apx-1-Minimality *returns an instance* $\widetilde{\mathcal{I}}$ *at Line 15. Then,* $\widetilde{\mathcal{I}}$ *is a* 1*-minimal instance with no empty constraint and* $\boldsymbol{w}_{\widetilde{\mathcal{I}}} \geq 1 - \epsilon_1$.

Proof. Suppose that we reach Line 15 in the t-th iteration. First, we note that $t \leq \frac{K|D|}{\epsilon_2}$ from Lemma D.10. If we do not return at Line 15 and continue process, the instance $\mathcal{I}^{t+1}$ will have no empty constraint from Lemma D.12.

Fix a constraint $C = (S, R)$ in $\mathcal{I}^0$. Abusing notations, we use the same C to denote the corresponding constraint in $\mathcal{I}^i$ for $i \leq t$. We associate a set $\mathcal{S}_C = \{(\{v\}, \mathrm{pr}_v R) \mid v \in S\}$ with C and consider how $|\mathcal{S}_C|$ decreases. In the beginning, $|\mathcal{S}_C| \leq K|D|$ clearly holds. Suppose that $\mathcal{D}^i_{\mathcal{I}}$ contains C for some i. Then, $|\mathcal{S}_C|$ decreases at least by one. Thus, if $\#\{i \in [t] \mid C \in \mathcal{D}^i_{\mathcal{I}}\} \geq K|D| - 1$, $|\mathcal{S}_C|$ contains at most one element. It follows that C contains exactly one tuple in $\mathcal{I}^t$ since $\mathcal{I}^{t+1}$ has no empty constraint.

We have $\sum_{i=1}^{t} |\mathcal{D}^i_{\mathcal{I}}| \geq K|D|\boldsymbol{w}_{\mathcal{I}^0}$. Thus, every constraint in $\mathcal{C}^t_{\mathcal{I}}$ contains exactly one tuple. Then, from Lemma D.8, $\widetilde{\mathcal{I}}$ is 1-minimal and has no empty constraint. Also, $\boldsymbol{w}_{\widetilde{\mathcal{I}}} \geq \boldsymbol{w}_{\mathcal{I}^0} \geq 1 - \epsilon_1$. □

Proof of Theorem D.1. Suppose that Apx-1-Minimality terminates at Line 13 in the t-th iteration. Let $\mathcal{D}'_{\mathcal{I}}$ be the set of constraints in $\mathcal{I}^0$ corresponding to $\mathcal{D}^t_{\mathcal{I}}$. We consider an instance $\mathcal{I}' = (V, D, \mathcal{C}^0_{\mathcal{I}} \setminus \mathcal{D}'_{\mathcal{I}}, \boldsymbol{w})$. Note that $\boldsymbol{w}_{\mathcal{I}'} \geq 1 - \epsilon_1 - \epsilon_2 \geq 1 - O(\frac{K|D|\log(K|D|)}{\log(1/2\epsilon)})$ from Lemma D.13. We will show that $\mathcal{I}'$ has a satisfying assignment α. Then, it is clear that $\mathrm{Val}_{\mathcal{I}}(\alpha) \geq 1 - O(\frac{K|D|\log(K|D|)}{\log(1/2\epsilon)})$.

Let $\mathcal{I}''$ be the instance obtained from $\mathcal{I}'$ by applying 1-Minimality. Suppose, for contradiction, that $\mathcal{I}''$ has an empty constraint. Note that the empty constraint is originated from $\mathcal{C}^0_{\mathcal{I}} \setminus \mathcal{D}'_{\mathcal{I}}$. Then, since $\mathcal{I}^0$ contains all the constraints in $\mathcal{I}'$, $\widetilde{\mathcal{I}}$ also must have an empty constraint, which is a contradiction. Thus, $\mathcal{I}''$ has no empty constraint, and it follows that $\mathcal{I}'$ has a satisfying assignment since $\mathcal{I}'$ is an instance of a CSP of width 1.

From the same argument and Lemma D.14, when Apx-1-Minimality terminates at Line 15, we can obtain an assignment α such that $\mathrm{Val}_{\mathcal{I}}(\alpha) \geq 1 - O(\frac{K|D|\log(K|D|)}{\log(1/2\epsilon)})$. □