SIAM J. COMPUT.
Vol. 48, No. 6, pp. 1763–1795

ROBUST ALGORITHMS WITH POLYNOMIAL LOSS FOR NEAR-UNANIMITY CSPs*

VÍCTOR DALMAU†, MARCIN KOZIK‡, ANDREI KROKHIN§, KONSTANTIN MAKARYCHEV¶, YURY MAKARYCHEV‖, AND JAKUB OPRŠAL§

Abstract. An instance of the constraint satisfaction problem (CSP) is given by a family of constraints on overlapping sets of variables, and the goal is to assign values from a fixed domain to the variables so that all constraints are satisfied. In the optimization version, the goal is to maximize the number of satisfied constraints. An approximation algorithm for a CSP is called robust if it outputs an assignment satisfying an $(1-g(\varepsilon))$-fraction of constraints on any $(1-\varepsilon)$-satisfiable instance, where the loss function g is such that $g(\varepsilon)\to 0$ as $\varepsilon\to 0$. We study how the robust approximability of CSPs depends on the set of constraint relations allowed in instances, the so-called constraint language. All constraint languages admitting a robust polynomial-time algorithm (with some g) have been characterized by Barto and Kozik, with the general bound on the loss g being doubly exponential, specifically $g(\varepsilon)=O((\log\log(1/\varepsilon))/\log(1/\varepsilon))$. It is natural to ask when a better loss can be achieved, in particular polynomial loss $g(\varepsilon)=O(\varepsilon^{1/k})$ for some constant k. In this paper, we consider CSPs with a constraint language having a near-unanimity polymorphism. This general condition almost matches a known necessary condition for having a robust algorithm with polynomial loss. We give two randomized robust algorithms with polynomial loss for such CSPs: one works for any near-unanimity polymorphism and the parameter k in the loss depends on the size of the domain and the arity of the relations in Γ, while the other works for a special ternary near-unanimity operation called the dual discriminator with $k=2$ for any domain size. In the latter case, the CSP is a common generalization of UNIQUE GAMES with a fixed domain and 2-SAT. In the former case, we use the algebraic approach to the CSP. Both cases use the standard semidefinite programming relaxation for the CSP.

Key words. constraint satisfaction, approximation algorithms, robust algorithm, near-unanimity polymorphism

AMS subject classifications. 68Q17, 68W20, 68W25, 68W40

DOI. 10.1137/18M1163932

1. Introduction.

The constraint satisfaction problem (CSP) provides a framework in which it is possible to express, in a natural way, many combinatorial problems encountered in computer science and AI [18, 20, 25]. An instance of the CSP consists of a set of variables, a domain of values, and a set of constraints on combinations of

*Received by the editors January 4, 2018; accepted for publication (in revised form) September 4, 2019; published electronically November 26, 2019. A preliminary version of this paper appeared in SODA 2017.

https://doi.org/10.1137/18M1163932

Funding: The first author's research was partially supported by MINECO under grant TIN2016-76573-C2-1-P and Maria de Maeztu Units of Excellence programme MDM-2015-0502. The fifth author's research was partially supported by NSF awards CAREER CCF-1150062 and IIS-1302662. The sixth author's research was supported by the European Research Council (grant agreement 681988, CSP-Infinity). The research of the second and sixth authors was partially supported by the National Science Centre Poland under grant UMO-2014/13/B/ST6/01812.

†Department of Information and Communication Technologies, University Pompeu Fabra, Barcelona, 08018, Spain (victor.dalmau@upf.edu).

‡Department of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, 30-348, Poland (marcin.kozik@uj.edu.pl).

§Department of Computer Science, Durham University, Durham, DH1 3LE, UK (andrei.krokhin@durham.ac.uk, jakub.oprsal@durham.ac.uk).

¶Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 (konstantin@northwestern.edu).

‖Toyota Technological Institute at Chicago, Chicago, IL 60637 (yury@ttic.edu).

values that can be taken by certain subsets of variables. The basic aim is then to find an assignment of values to the variables that satisfies the constraints (decision version) or that satisfies the maximum number of constraints (optimization version).

Since CSP-related algorithmic tasks are usually hard in full generality, a major line of research in CSP studies how possible algorithmic solutions depend on the set of relations allowed to specify constraints, the so-called *constraint language* (see, e.g., [11, 18, 20, 25, 42]). The constraint language is denoted by Γ and the corresponding CSP by CSP(Γ). For example, when one is interested in polynomial-time solvability (to optimality for the optimization case), the ultimate sort of results are dichotomy results [10, 11, 25, 38, 50, 52], pioneered in [49], which characterize the tractable restrictions and show that the rest are NP-hard. Classifications with respect to other complexity classes or specific algorithms are also of interest (see, e.g., [4, 6, 39, 44]). When approximating (optimization) CSPs, the goal is to improve, as much as possible, the quality of approximation that can be achieved in polynomial time; see, e.g., surveys [35, 47]. Throughout the paper, we assume that P$\neq$NP.

The study of *almost satisfiable* CSP instances features prominently in the approximability literature. On the hardness side, the notion of approximation resistance (which, intuitively, means that a problem cannot be approximated better than by just picking a random assignment, even on almost satisfiable instances) has been much studied recently; see, e.g., [1, 15, 29, 37]. Many exciting developments in approximability in the last decade were driven by the *unique games conjecture* (UGC) of Khot; see survey [35]. The UGC states that it is NP-hard to tell almost satisfiable instances of CSP(Γ) from those where only a small fraction of constraints can be satisfied, where Γ is the constraint language consisting of all graphs of permutations over a large enough domain. This conjecture (if true) is known to imply optimal inapproximability results for many classical optimization problems [35]. Moreover, if the UGC is true, then a simple algorithm based on semidefinite programming (SDP) provides the best possible approximation for all optimization problems CSP(Γ) [48], though the exact quality of this approximation is unknown.

On the positive side, Zwick [53] initiated the systematic study of approximation algorithms which, given an almost satisfiable instance, find an almost satisfying assignment. Formally, call a polynomial-time algorithm for a CSP *robust* if, for every $\varepsilon > 0$ and every $(1-\varepsilon)$-satisfiable instance (i.e., at most a ε-fraction of constraints can be removed to make the instance satisfiable), it outputs a $(1-g(\varepsilon))$-satisfying assignment (i.e., that fails to satisfy at most a $g(\varepsilon)$-fraction of constraints). Here, the *loss* function g must be such that $g(\varepsilon) \to 0$ as $\varepsilon \to 0$. Note that one can without loss of generality assume that $g(0) = 0$, that is, a robust algorithm must return a satisfying assignment for any satisfiable instance. The running time of the algorithm should not depend on ε (which is unknown when the algorithm is run). Which problems CSP(Γ) admit robust algorithms? When such algorithms exist, how does the best possible loss g depend on Γ?

Related work. In [53], Zwick gave an SDP-based robust algorithm with $g(\varepsilon) = O(\varepsilon^{1/3})$ for 2-SAT and an LP-based robust algorithm with $g(\varepsilon) = O(1/\log(1/\varepsilon))$ for HORN k-SAT. Robust algorithms with $g(\varepsilon) = O(\sqrt{\varepsilon})$ were given in [17] for 2-SAT and in [16] for UNIQUE GAMES(q), where q denotes the size of the domain. For HORN-2-SAT, a robust algorithm with $g(\varepsilon) = 2\varepsilon$ was given in [27]. These bounds for HORN k-SAT ($k \geq 3$), HORN 2-SAT, 2-SAT, and UNIQUE GAMES(q) are known to be optimal [27, 34, 36], assuming the UGC.

The algebraic approach to the CSP [11, 18, 32] has played a significant role in the

recent massive progress in understanding the landscape of complexity of CSPs. The key to this approach is the notion of a *polymorphism*, which is an n-ary operation (on the domain) that preserves the constraint relations. Intuitively, a polymorphism provides a uniform way to combine n solutions to a system of constraints (say, part of an instance) into a new solution by applying the operation componentwise. The intention is that the new solution improves on the initial solutions in some problem-specific way. Many classifications of CSPs with respect to some algorithmic property of interest begin by proving an algebraic classification stating that every constraint language either can simulate (in a specific way, via gadgets; see, e.g., [5, 23, 44] for details) one of a few specific basic CSPs failing the property of interest or else has polymorphisms having certain nice properties (say, satisfying nice equations). Such polymorphisms are then used to obtain positive results, e.g., to design and analyze algorithms. Getting such a positive result in full generality in one step is usually hard, so (typically) progress is made through a series of intermediate steps where the result is obtained for increasingly weaker algebraic conditions. The algebraic approach was originally developed for the decision CSP [11, 32], and it was adapted for robust satisfiability in [23].

One such algebraic classification result [45] gives an algebraic condition (referred to as SD($\wedge$) or "omitting types **1** and **2**"; see [4, 41, 45] for details) equivalent to the *in*ability to simulate 3-Lin-p—systems of linear equations over Z_p, p prime, with three variable per equation. Håstad's celebrated result [28] implies that 3-Lin-p does not admit a robust algorithm (for any g). This result carries over to all constraint languages that can simulate (some) 3-Lin-p [23]. The remaining languages are precisely those that have the logico-combinatorial property of CSPs called "*bounded width*" or "*bounded treewidth duality*" [4, 9, 46]. This property says, roughly, that all unsatisfiable instances can be refuted via local propagation; see [12] for a survey on dualities for the CSP. Barto and Kozik used SD($\wedge$) in [4], and then in [5] they used their techniques from [4] to prove the Guruswami–Zhou conjecture [27] that each bounded width CSP admits a robust algorithm.

The general bound on the loss in [5] is $g(\varepsilon) = O((\log\log(1/\varepsilon))/\log(1/\varepsilon))$. It is natural to ask when a better loss can be achieved. In particular, the problems of characterizing CSPs where linear loss $g(\varepsilon) = O(\varepsilon)$ or polynomial loss $g(\varepsilon) = O(\varepsilon^{1/k})$ (for constant k) can be achieved have been posed in [23]. Partial results on these problems appeared in [23, 24, 43]. For the Boolean case, i.e., when the domain is $\{0, 1\}$, the dependence of loss on Γ is fully classified in [23].

Our contribution. We study CSPs that admit a robust algorithm with polynomial loss. As explained above, the bounded width property is necessary for admitting any robust algorithm. Horn 3-Sat has bounded width but does not admit a robust algorithm with polynomial loss (unless the UGC fails) [27]. The algebraic condition that separates 3-Lin-p and Horn 3-Sat from the CSPs that can potentially be shown to admit a robust algorithm with polynomial loss is known as SD($\vee$) or "omitting types **1**, **2**, and **5**" [23]; see section 2.2 for the description of SD($\vee$) in terms of polymorphisms. The condition SD($\vee$) is also a necessary condition for the logico-combinatorial property of CSPs called "*bounded pathwidth duality*" (which says, roughly, that all unsatisfiable instances can be refuted via local propagation in a linear fashion) and possibly a sufficient condition for it too [44]. It seems very hard to obtain a robust algorithm with polynomial loss for every CSP satisfying SD($\vee$) all in one step.

From the algebraic perspective, the most general natural condition that is (slightly)

stronger than SD($\vee$) is the *near-unanimity (NU)* condition [2]. CSPs with a constraint language having an NU polymorphism received a lot of attention in the literature (see, e.g., [25, 31, 6]). Bounded pathwidth duality for CSPs admitting an NU polymorphism was established in a series of papers [21, 22, 6], and we use some ideas from [22, 6] in this paper.

We prove that any CSP with a constraint language having an NU polymorphism admits a randomized robust algorithm with loss $O(\varepsilon^{1/k})$, where k depends on the size of the domain. It is an open question whether this dependence on the size of the domain is necessary. We prove that, for the special case of a ternary NU polymorphism known as the *dual discriminator* (the corresponding CSP is a common generalization of Unique Games with a fixed domain and 2-Sat), we can always choose $k = 2$. Like the vast majority of approximation algorithms for CSPs [47], our algorithms use the standard SDP relaxation.

The algorithm for the general NU case follows the same general scheme as [5, 43]:

1. Solve the LP/SDP relaxation for a $(1 - \varepsilon)$-satisfiable instance $\mathcal{I}$.
2. Use the LP/SDP solution to remove certain constraints in $\mathcal{I}$ with total weight $O(g(\varepsilon))$ (in our case, $O(\varepsilon^{1/k})$) so that the remaining instance satisfies a certain consistency condition.
3. Use the appropriate polymorphism (in our case, NU) to show that any instance of CSP(Γ) with this consistency condition is satisfiable.

Steps 1 and 2 in this scheme can be applied to any CSP instance, and this is where essentially all work of the approximation algorithm happens. Polymorphisms are not used in the algorithm; they are used in step 3 only to prove the correctness. While the above general scheme is rather simple, applying it is typically quite challenging. Obviously, step 2 prefers weaker conditions (achievable by removing not too many constraints), while step 3 prefers stronger conditions (so that they can guarantee satisfiability), so reaching the balance between them is the main (and typically significant) technical challenge in any application of this scheme. Our algorithm is somewhat inspired by [5], but it is also quite different from the algorithm there. That algorithm is designed so that steps 1 and 2 establish a consistency condition that, in particular, includes the 1-minimality condition, and establishing 1-minimality alone requires removing constraints with total weight $O(1/\log(1/\varepsilon))$ [27], unless UGC fails. Since our requirement on the loss function $g(\varepsilon)$ is stricter, we have to design a different "rounding" procedure (which is usually the hardest part to analyze for most approximation algorithms). As in [5], our rounding is nontraditional, since a solution to the SDP relaxation is used to decide which constraints to violate, rather than to immediately assign values to the variables. To show that our rounding gives the right dependency on ε, we introduce a new consistency condition somewhat inspired by [6, 40]. The proof that the new consistency condition satisfies the requirements of steps 2 and 3 of the above scheme is one of the main technical contributions of our paper.

Organization of the paper. After some preliminaries, we formulate the two main results of this paper in section 3. Section 4 then contains a description of SDP relaxations that we will use later. Sections 5 and 6 contain the description of the algorithms for constraint languages compatible with NU polymorphism and the dual discriminator, respectively; the following sections prove the correctness of the two algorithms.

2. Preliminaries.

2.1. CSPs. Throughout the paper, let D be a *fixed finite* set, sometimes called the *domain*. An *instance* of the CSP is a pair $\mathcal{I} = (V, \mathcal{C})$ with V a finite set of *variables* and $\mathcal{C}$ a finite set of constraints. Each constraint is a pair $(\overline{x}, R)$, where $\overline{x}$ is a tuple of variables (say, of length $r > 0$) called the *scope* of C, and R is an r-ary relation on D called the *constraint relation* of C. The arity of a constraint is defined to be the arity of its constraint relation. In the weighted optimization version, which we consider in this paper, every constraint $C \in \mathcal{C}$ has an associated *weight* $w_C \geq 0$. Unless otherwise stated, we shall assume that every instance satisfies $\sum_{C \in \mathcal{C}} w_C = 1$.

An *assignment* for $\mathcal{I}$ is a mapping $s\colon V \to D$. We say that s satisfies a constraint $((x_1, \ldots, x_r), R)$ if $(s(x_1), \ldots, s(x_r)) \in R$. For $0 \leq \beta \leq 1$, we say that assignment s β-satisfies $\mathcal{I}$ if the total weight of the constraints satisfied by s is at least β. In this case, we say that $\mathcal{I}$ is β-satisfiable. The best possible β for $\mathcal{I}$ is denoted by $\mathrm{Opt}(\mathcal{I})$.

A *constraint language* on D is a *finite* set Γ of relations on D. The problem $\mathrm{CSP}(\Gamma)$ consists of all instances of the CSP where all the constraint relations are from Γ. Problems k-Sat, Horn k-Sat, 3-Lin-p, Graph H-coloring, and Unique Games($|D|$) are all of the form $\mathrm{CSP}(\Gamma)$.

The *decision problem* for $\mathrm{CSP}(\Gamma)$ asks whether an input instance $\mathcal{I}$ of $\mathrm{CSP}(\Gamma)$ has an assignment satisfying all constraints in $\mathcal{I}$. The *optimization problem* for $\mathrm{CSP}(\Gamma)$ asks one to find an assignment s where the weight of the constraints satisfied by s is as large as possible. Optimization problems are often hard to solve to optimality, motivating the study of *approximation* algorithms.

2.2. Algebra. An n-ary operation f on D is a map from D^n to D. We say that f *preserves* (or is a *polymorphism* of) an r-ary relation R on D if for all n (not necessarily distinct) tuples $(a_1^i, \ldots, a_r^i) \in R$, $1 \leq i \leq n$, the tuple

$$(f(a_1^1, \ldots, a_n^1), \ldots, f(a_1^r, \ldots, a_n^r))$$

belongs to R as well. If R is the edge relation of a digraph H, then f is a polymorphism of R if and only if, for any list of n (not necessarily distinct) edges $(a_1, b_1), \ldots, (a_n, b_n)$ of H, there is an edge in H from $f(a_1, \ldots, a_n)$ to $f(b_1, \ldots, b_n)$. If f is a polymorphism of every relation in a constraint language Γ, then f is called a polymorphism of Γ. Many algorithmic properties of $\mathrm{CSP}(\Gamma)$ depend only on the polymorphisms of Γ; see survey [7], as well as [11, 23, 32, 44].

An $(n+1)$-ary ($n \geq 2$) operation f is a near-unanimity (NU) operation if, for all $x, y \in D$, it satisfies

$$f(x, x, \ldots, x, x, y) = f(x, x, \ldots, x, y, x) = \cdots = f(y, x, \ldots, x, x, x) = x.$$

Note that the behavior of f on other tuples of arguments is not restricted. An NU operation of arity 3 is called a *majority* operation.

We mentioned in the introduction that (modulo UGC) only constraint languages satisfying condition SD($\vee$) can admit robust algorithms with polynomial loss. The condition SD($\vee$) can be expressed in many equivalent ways, for example as the existence of ternary polymorphisms $d_0, \ldots, d_t$, $t \geq 2$, satisfying the following equations [30]:

$$d_0(x, y, z) = x, \quad d_t(x, y, z) = z, \tag{2.1}$$

$$d_i(x, y, x) = d_{i+1}(x, y, x) \text{ for all even } i < t, \tag{2.2}$$

$$d_i(x, y, y) = d_{i+1}(x, y, y) \text{ for all even } i < t, \tag{2.3}$$

$$d_i(x, x, y) = d_{i+1}(x, x, y) \text{ for all odd } i < t. \tag{2.4}$$

If line (2.2) is strengthened to $d_i(x, y, x) = x$ for all i, then, for any constraint language, having such polymorphisms would be equivalent to having an NU polymorphism of some arity [3] (this is true only when constraint languages are assumed to be finite).

NU polymorphisms have appeared many times in the CSP literature. For example, they characterize the so-called bounded strict width property [25, 31], which says, roughly, that, after establishing local consistency in an instance, one can always construct a solution in a greedy way, by picking values for variables in any order so that constraints are not violated.

THEOREM 2.1 (see [25, 31]). *Let Γ be a constraint language with an NU polymorphism of some arity. There is a polynomial-time algorithm that, given an instance of* CSP(Γ)*, finds a satisfying assignment or reports that none exists.*

Every relation with an $(n+1)$-ary NU polymorphism is *n-decomposable* (and in some sense the converse also holds) [2]. We give a formal definition only for the majority case $n = 2$. Let R be an r-ary ($r \geq 2$) relation. For every $i, j \in \{1, \dots, r\}$, let $\mathrm{pr}_{i,j} R$ be the binary relation $\{(a_i, a_j) \mid (a_1, \dots, a_r) \in R\}$. Then R is called 2-*decomposable* if the following holds: a tuple $(a_1, \dots, a_r) \in D^r$ belongs to R if and only if $(a_i, a_j) \in \mathrm{pr}_{i,j} R$ for every $i, j \in \{1, \dots, r\}$.

The *dual discriminator* is a majority operation f such that $f(x, y, z) = x$ whenever x, y, z are pairwise distinct. Binary relations preserved by the dual discriminator are known as *implicational* [8] or *0/1/all* [19] relations. Every such relation is of one of the four following types:

1. $(\{a\} \times D) \cup (D \times \{b\})$ for $a, b \in D$;
2. $\{(\pi(a), a) \mid a \in D\}$, where π is a permutation on D;
3. $P \times Q$, where $P, Q \subseteq D$;
4. an intersection of a relation of type 1 or 2 with a relation of type 3.

The relations of the first kind, when $D = \{0, 1\}$, are exactly the relations allowed in 2-SAT, while the relations of the second kind are precisely the relations allowed in UNIQUE GAMES ($|D|$). We remark that having such an explicit description of relations having a given polymorphism is rare beyond the Boolean case.

3. Main result.

THEOREM 3.1. *Let Γ be a constraint language on D:*

(1) *If Γ has an NU polymorphism, then* CSP(Γ) *admits a randomized polynomial-time robust algorithm with loss $O(\varepsilon^{1/k})$ for $k = 6|D|^r + 7$, where r is the maximal arity of a relation in Γ. Moreover, if Γ contains only binary relations, then one can choose $k = 6|D| + 7$.*

(2) *If Γ has the dual discriminator polymorphism, then* CSP(Γ) *admits a randomized polynomial-time robust algorithm with loss $O(\sqrt{\varepsilon})$.*

It was stated as an open problem in [23] whether every CSP that admits a robust algorithm with loss $O(\varepsilon^{1/k})$ admits one where k is bounded by an absolute constant (that does not depend on D). In the context of the above theorem, the problem can be made more specific: is dependence of k on $|D|$ in this theorem avoidable, or is there a strict hierarchy of possible degrees there? The case of a majority polymorphism is a good starting point when trying to answer this question.

As mentioned in the introduction, robust algorithms with polynomial loss and bounded pathwidth duality for CSPs seem to be somehow related, at least in terms of algebraic conditions. The condition SD($\vee$) is the common necessary condition for them, although it is conditional on the UGC for the former and unconditional for

the latter. Having an NU polymorphism is a sufficient condition for both. Another family of problems CSP(Γ) with bounded pathwidth duality was shown to admit robust algorithms with polynomial loss in [23], where the parameter k depends on the pathwidth duality bound (and appears in the algebraic description of this family). This family includes languages not having an NU polymorphism of any arity; see [13, 14]. It is unclear how far connections between the two directions go, but consistency notions seem to be the common theme.

Returning to the discussion of a possible hierarchy of degrees in polynomial loss in robust algorithms, there was a similar question about a hierarchy of bounds for pathwidth duality, and the hierarchy was shown to be strict [22], even in the presence of a majority polymorphism.

4. SDP relaxation. Associated to every instance $\mathcal{I} = (V, \mathcal{C})$ of the CSP there is a standard SDP relaxation. It comes in two versions: maximizing the number of satisfied constraints and minimizing the number of unsatisfied constraints. We use the latter. We define it assuming that all constraints are binary; this will be sufficient for our purposes. The SDP has a variable $\mathbf{x}_a$ for every $x \in V$ and $a \in D$. It also contains a special unit vector $\mathbf{v}_0$. The goal is to assign $(|V\|D|)$-dimensional real vectors to its variables minimizing the following objective function:

$$\sum_{C=((x,y),R)\in\mathcal{C}} w_C \sum_{(a,b)\notin R} \mathbf{x}_a \mathbf{y}_b \tag{4.1}$$

subject to

$$\mathbf{x}_a \mathbf{y}_b \geq 0, \qquad x, y \in V, a, b \in D, \tag{4.2}$$

$$\mathbf{x}_a \mathbf{x}_b = 0, \qquad x \in V, a, b \in D, a \neq b, \tag{4.3}$$

$$\textstyle\sum_{a\in D} \mathbf{x}_a = \mathbf{v}_0, \qquad x \in V, \tag{4.4}$$

$$\|\mathbf{v}_0\| = 1. \tag{4.5}$$

In the intended integral solution, $x = a$ if $\mathbf{x}_a = \mathbf{v}_0$. In the fractional solution, we informally interpret $\|\mathbf{x}_a\|^2$ as the probability of $x = a$ according to the SDP (the constraints of the SDP ensure that $\sum_{a\in D} \|\mathbf{x}_a\|^2 = 1$). If $C = ((x, y), R)$ is a constraint and $a, b \in D$, one can think of $\mathbf{x}_a \mathbf{y}_b$ as the probability given by the solution of the SDP to the pair (a, b) in C. The optimal SDP solution, then, gives as little probability as possible to pairs that are not in the constraint relation. For a constraint $C = ((x, y), R)$, conditions (4.4) and (4.5) imply that $\sum_{(a,b)\in R} \mathbf{x}_a \mathbf{y}_b$ is at most 1. Let $\mathrm{loss}(C) = \sum_{(a,b)\notin R} \mathbf{x}_a \mathbf{y}_b$. For a subset $A \subseteq D$, let $\mathbf{x}_A = \sum_{a\in A} \mathbf{x}_a$. Note that $\mathbf{x}_D = \mathbf{y}_D (= \mathbf{v}_0)$ for all $x, y \in D$.

Let $\mathrm{SDPOpt}(\mathcal{I})$ be the optimum value of (4.1). It is clear that, for any instance $\mathcal{I}$, we have $\mathrm{Opt}(\mathcal{I}) \geq \mathrm{SDPOpt}(\mathcal{I}) \geq 0$. There are algorithms [51] that, given an SDP instance $\mathcal{I}$ and some additive error $\delta > 0$, produce in time $poly(|\mathcal{I}|, \log(1/\delta))$ an output vector solution whose value is at most $\mathrm{SDPOpt}(\mathcal{I}) + \delta$. There are several ways to deal with the error δ. In this paper, we deal with it by introducing a preprocessing step which will also be needed to argue that the algorithm described in the proof of Theorem 3.1(1) runs in polynomial time.

Preprocessing step 1. Assume that $\mathcal{C} = \{C_1, \ldots, C_m\}$ and that $w_{C_1} \geq w_{C_2} \geq \cdots \geq w_{C_m}$. Using the algorithm from Theorem 2.1, find the largest j such that the subinstance $\mathcal{I}_j = (V, \{C_1, \ldots, C_j\})$ is satisfiable. If the total weight of the constraints in $\mathcal{I}_j$ is at least $1 - 1/m$, then return the assignment s satisfying $\mathcal{I}_j$ and stop.

LEMMA 4.1. *Assume that $\mathcal{I}$ is $(1-\varepsilon)$-satisfiable. If $\varepsilon \le 1/m^2$, then preprocessing step 1 returns an assignment that $(1-\sqrt{\varepsilon})$-satisfies $\mathcal{I}$.*

Proof. Assume $\varepsilon \le 1/m^2$. Let i be maximum with the property that $w_{C_i} > \varepsilon$. It follows that the instance $\mathcal{I}_i = (V, \{C_1, \dots, C_i\})$ is satisfiable since the assignment $(1-\varepsilon)$-satisfying $\mathcal{I}$ must satisfy every constraint with weight larger than ε. It follows that $i \le j$ and, hence, the value of the assignment satisfying $\mathcal{I}_j$ is at least $1 - w_{C_{i+1}} - \cdots - w_{C_m} \ge 1 - m w_{C_{i+1}} \ge 1 - m\varepsilon \ge 1 - \sqrt{\varepsilon}$. ∎

If the preprocessing step returns an assignment, then we are done. So assume that it did not return an assignment. Then we know that $\varepsilon \ge 1/m^2$. We then solve the SDP relaxation with $\delta = 1/m^2$ obtaining a solution with objective value at most 2ε, which is good enough for our purposes.

5. Overview of the proof of Theorem 3.1(1). We assume throughout that Γ has an NU polymorphism of arity $n+1$ ($n \ge 2$).

It is sufficient to prove Theorem 3.1(1) for the case when Γ consists of binary relations and $k = 6|D| + 7$. The rest will follow by Proposition 4.1 of [5] (see also Theorem 24 in [7]), which shows how to reduce the general case to constraint languages consisting of unary and binary relations in such a way that the domain size increases from $|D|$ to $|D|^r$, where r is the maximal arity of a relation in Γ. Note that every unary constraint (x, R) can be replaced by the binary constraint $((x,x), R')$, where $R' = \{(a,a) \mid a \in R\}$.

Throughout the rest of this section, let $\mathcal{I} = (V, \mathcal{C})$ be a $(1-\varepsilon)$-satisfiable instance of CSP(Γ).

5.1. Patterns and realizations. A *pattern in $\mathcal{I}$* is defined as a directed multigraph p whose vertices are labeled by variables of $\mathcal{I}$ and edges are labeled by constraints of $\mathcal{I}$ in such a way that the beginning of an edge labeled by $((x,y), R)$ is labeled by x and the end by y. Two of the vertices in p can be distinguished as the *beginning* and the *end* of p. If these two vertices are labeled by variables x and y, respectively, then we say that p is a pattern from x to y.

For two patterns p and q such that the end of p and the beginning of q are labeled by the same variable, we define $p+q$ to be the pattern which is obtained from the disjoint union of p and q by identifying the end of p with the beginning of q and choosing the beginning of $p+q$ to be the beginning of p and the end of $p+q$ to be the end of q. We also define jp to be $p+\cdots+p$, where p appears j times. A pattern is said to be a *path pattern* if the underlying graph is an oriented path with the beginning and the end being the two end vertices of the path, and it is said to be an *n-tree pattern* if the underlying graph is an orientation of a tree with at most n leaves and both the beginning and the end are leaves. A *path of n-trees pattern* is then any pattern of the form $t_1 + \cdots + t_j$ for some n-tree patterns $t_1, \dots, t_j$.

A *realization of a pattern p* is a mapping r from the set of vertices of p to D such that if (v_x, v_y) is an edge labeled by $((x,y), R)$, then $(r(v_x), r(v_y)) \in R$. Note that r does not have to map different vertices of p labeled with same variable to the same element in D. A *propagation* of a set $A \subseteq D$ along a pattern p whose beginning vertex is b and ending vertex is e is defined as follows. For $A \subseteq D$, define $A+p = \{r(e) \mid \ r \text{ is a realization of } p \text{ with } r(b) \in A\}$. Also, for a binary relation R, we put $A+R = \{b \mid (a,b) \in R \text{ and } a \in A\}$. Observe that we have $(A+p)+q = A+(p+q)$.

Further, assume that we have nonempty sets D_x^ℓ, where $1 \le \ell \le |D|+1$ and x runs through all variables in an instance $\mathcal{I}$. Let p be a pattern in $\mathcal{I}$ with the beginning b and the end e. We call a realization r of p an *ℓ-realization* (with respect to the family $\{D_x^\ell\}$) if, for any vertex v of p labeled by a variable x, we have $r(v) \in D_x^{\ell+1}$. For

$A \subseteq D$, define

$$A +^{\ell} p = \{r(e) \mid r \text{ is an } \ell\text{-realization of } p \text{ with } r(b) \in A\}.$$

Also, for a constraint $((x,y),R)$ or $((y,x),R^{-1})$ and sets $A, B \subseteq D$, we write $B = A +^{\ell} (x,R,y)$ if $B = \{b \in D_y^{\ell+1} \mid (a,b) \in R \text{ for some } a \in A \cap D_x^{\ell+1}\}$.

5.2. The consistency notion. Recall that we assume that Γ contains only binary relations. Before we formally introduce the new consistency notion, which is the key to our result, as we explained in the introduction, we give an example of a similar simpler condition. We mentioned before that 2-SAT is a special case of a CSP that admits an NU polymorphism (actually, the only majority operation on $\{0,1\}$). There is a textbook consistency condition characterizing satisfiable 2-SAT instances, which can be expressed in our notation as follows: for each variable x in a 2-SAT instance $\mathcal{I}$, there is a value a_x such that, for any path pattern p in $\mathcal{I}$ from x to x, we have $a_x \in \{a_x\} + p$.

Let $\mathcal{I}$ be an instance of CSP(Γ) over a set V of variables. We say that $\mathcal{I}$ satisfies condition $(\text{IPQ})_n$ if the following holds:

$(\text{IPQ})_n$ For every $y \in V$, there exist nonempty sets $D_y^1 \subseteq \cdots \subseteq D_y^{|D|} \subseteq D_y^{|D|+1} = D$ such that for any $x \in V$, any $\ell \le |D|$, any $a \in D_x^{\ell}$, and any two patterns p, q which are paths of n-trees in $\mathcal{I}$ from x to x, there exists j such that

$$a \in \{a\} +^{\ell} (j(p+q)+p).$$

Note that $+$ between p and q is the pattern addition and thus is independent of ℓ. Note also that a in the above condition belongs to D_x^{ℓ}, while propagation is performed by using ℓ-realizations, i.e., inside sets $D_y^{\ell+1}$.

The following theorem states that this consistency notion satisfies the requirements of step 3 of the general scheme (for designing robust approximation algorithms) discussed in the introduction.

THEOREM 5.1. *Let Γ be a constraint language containing only binary relations such that Γ has an $(n+1)$-ary NU polymorphism. If an instance $\mathcal{I}$ of* CSP(Γ) *satisfies $(\text{IPQ})_n$, then $\mathcal{I}$ is satisfiable.*

5.3. The algorithm. Let $k = 6|D| + 7$. We provide an algorithm which, given a $(1-\varepsilon)$-satisfiable instance $\mathcal{I}$ of CSP(Γ), removes $O(\varepsilon^{1/k})$ constraints from it to obtain a subinstance $\mathcal{I}'$ satisfying condition $(\text{IPQ})_n$. It then follows from Theorem 5.1 that $\mathcal{I}'$ is satisfiable, and we can find a satisfying assignment by Theorem 2.1.

5.3.1. More preprocessing. By Lemma 4.1, we can assume that $\varepsilon \ge 1/m^2$. We solve the SDP relaxation with error $\delta = 1/m^2$ and obtain a solution $\{\mathbf{x}_a\}$ ($x \in V, a \in D$) whose objective value ε' is at most 2ε. Let us define α to be $\max\{\varepsilon', 1/m^2\}$. It is clear that $\alpha = O(\varepsilon)$. Furthermore, this gives us that $1/\alpha \le m^2$. This will be needed to argue that the main part of the algorithm runs in polynomial time.

Let $\kappa = 1/k$ (we will often use κ to avoid overloading formulas).

Preprocessing step 2. For each $x \in V$ and $1 \le \ell \le |D|+1$, compute sets $D_x^{\ell} \subseteq D$ as follows. Set $D_x^{|D|+1} = D$ and, for $1 \le \ell \le |D|$, set $D_x^{\ell} = \{a \in D \mid \|\mathbf{x}_a\| \ge r_{x,\ell}\}$, where $r_{x,\ell}$ is the smallest number of the form $r = \alpha^{3\ell\kappa}(2|D|)^{i/2}$, $i \ge 0$ integer, with $\{b \in D \mid r(2|D|)^{-1/2} \le \|\mathbf{x}_b\| < r\} = \emptyset$. It is easy to check that $r_{x,\ell}$ is obtained with $i \le |D|$.

It is clear that the sets $D_x^{\ell} \subseteq D$, $x \in V$, $1 \le \ell \le |D|$, can be computed in polynomial time.

The sets D_x^ℓ are chosen such that D_x^ℓ contains relatively "heavy" elements (a's such that $\|\mathbf{x}_a\|^2$ is large). The thresholds are chosen so that there is a big gap (at least by a factor of $2|D|$) between "heaviness" of an element in D_x^ℓ and outside.

5.3.2. Main part. Given that the preprocessing is done, we have that $1/\alpha \leq m^2$, and we precomputed sets D_x^ℓ for all $x \in V$ and $1 \leq \ell \leq |D|+1$. The description below uses the number n, where $n+1$ is the arity of the NU polymorphism of Γ.

Step 0. Remove every constraint C with $\mathrm{loss}(C) > \alpha^{1-\kappa}$.

Step 1. For every $1 \leq \ell \leq |D|$, do the following. Pick a value $r_\ell \in (0, \alpha^{(6\ell+4)\kappa})$ uniformly at random. Here we need some notation: for $x, y \in V$ and $A, B \subseteq D$, we write $\mathbf{x}_A \preceq^\ell \mathbf{y}_B$ to indicate that there is *no* integer j such that $\|\mathbf{y}_B\|^2 < r_\ell + j\alpha^{(6\ell+4)\kappa} \leq \|\mathbf{x}_A\|^2$. Then remove all constraints $((x,y),R)$ such that there are sets $A, B \subseteq D$ with $B = A +^\ell (x, R, y)$ and $\mathbf{x}_A \not\preceq^\ell \mathbf{y}_B$, or with $B = A +^\ell (y, R^{-1}, x)$ and $\mathbf{y}_A \not\preceq^\ell \mathbf{x}_B$.

Step 2. For every $1 \leq \ell \leq |D|$, do the following. Let $m_0 = \lfloor \alpha^{-2\kappa} \rfloor$. Pick a value $s_\ell \in \{0, \dots, m_0 - 1\}$ uniformly at random. We define $\mathbf{x}_A \preceq_w^\ell \mathbf{y}_B$ to mean that there is *no* integer j such that $\|\mathbf{y}_B\|^2 < r_\ell + (s_\ell + jm_0)\alpha^{(6\ell+4)\kappa} \leq \|\mathbf{x}_A\|^2$. Obviously, if $\mathbf{x}_A \preceq^\ell \mathbf{y}_B$, then $\mathbf{x}_A \preceq_w^\ell \mathbf{y}_B$. Now, if $A \subseteq B \subseteq D_x^{\ell+1}$ are such that $\|\mathbf{x}_B - \mathbf{x}_A\|^2 \leq (2n-3)\alpha^{(6\ell+4)\kappa}$ and $\mathbf{x}_B \not\preceq_w^\ell \mathbf{x}_A$, then remove all the constraints in which x participates.

Step 3. For every $1 \leq \ell \leq |D|$, do the following. Pick $m_\ell = \lceil \alpha^{-(3\ell+1)\kappa} \rceil$ unit vectors independently uniformly at random. For $x, y \in V$ and $A, B \subseteq D$, say that $\mathbf{x}_A$ and $\mathbf{y}_B$ are *cut* by a vector $\mathbf{u}$ if the signs of $\mathbf{u} \cdot (\mathbf{x}_A - \mathbf{x}_{D \setminus A})$ and $\mathbf{u} \cdot (\mathbf{y}_B - \mathbf{y}_{D \setminus B})$ differ. Furthermore, we say that $\mathbf{x}_A$ and $\mathbf{y}_B$ are ℓ-cut if they are cut by at least one of the chosen m_ℓ vectors. For every variable x, if there exist subsets $A, B \subseteq D$ such that $A \cap D_x^\ell \neq B \cap D_x^\ell$ and the vectors $\mathbf{x}_A$ and $\mathbf{x}_B$ are not ℓ-cut, then remove all the constraints in which x participates.

Step 4. For every $1 \leq \ell \leq |D|$, remove every constraint $((x,y),R)$ such that there are sets $A, B \subseteq D$ with $B = A +^\ell (x, R, y)$, and $\mathbf{x}_A$ and $\mathbf{y}_B$ are ℓ-cut, or with $B = A +^\ell (y, R^{-1}, x)$, and $\mathbf{y}_A$ and $\mathbf{x}_B$ are ℓ-cut.

Step 5. For every $1 \leq \ell \leq |D|$, do the following. For every variable x, if $A, B \subseteq D_x^{\ell+1}$ such that $\|\mathbf{x}_B - \mathbf{x}_A\|^2 \leq (2n-3)\alpha^{(6\ell+4)\kappa}$ and $\mathbf{x}_A$ and $\mathbf{x}_B$ are ℓ-cut, remove all constraints in which x participates.

Step 6. By Proposition 5.3 and Theorem 5.1, the remaining instance $\mathcal{I}'$ is satisfiable. Use the algorithm given by Theorem 2.1 to find a satisfying assignment for $\mathcal{I}'$. Assign all variables in $\mathcal{I}$ that do not appear in $\mathcal{I}'$ arbitrarily, and return the obtained assignment for $\mathcal{I}$.

Note that we chose to define the cut condition based on $\mathbf{x}_A - \mathbf{x}_{D \setminus A}$, rather than on $\mathbf{x}_A$, because the former choice has the advantage that $\|\mathbf{x}_A - \mathbf{x}_{D \setminus A}\| = 1$, which helps in some calculations.

In Step 0, we remove constraints that, according to the SDP solution, have a high probability to be violated. Intuitively, Steps 1 and 2 ensure that the loss in $\|\mathbf{x}_A\|$ after propagating A by a path of n-trees is not too big. This is achieved first by ensuring that by following a path we do not lose too much (Step 1), which also gives a bound on how much we can lose by following an n-tree pattern (see Lemma 8.13). Together with the removal of constraints in Step 2, this guarantees that following a path of n-trees we do not lose too much. This ensures that $\{a\} +^\ell (j(p+q)+p)$ is nonvanishing as j increases. Steps 3–5 ensure that if A and B are connected by paths of n-trees in both directions (i.e., $\mathbf{x}_A = \mathbf{x}_B +^\ell p_1$ and $\mathbf{x}_B = \mathbf{x}_A +^\ell p_2$), then $\mathbf{x}_A$ and $\mathbf{x}_B$ do not differ too much (i.e., $A \cap D_x^\ell = B \cap D_x^\ell$). This is achieved by separating

the space into cones by cutting it using the m_ℓ chosen vectors, removing the variables which have two different sets that are not ℓ-cut (Step 3), and then ensuring that if we follow an edge (Step 4), or if we drop elements that do not extend to an n-tree (Step 5), then we do not cross a border to another cone. This gives us both that the sequence $A_j = \{a\} +^\ell (j(p+q)+p)$ stabilizes and that, after it stabilizes, A_j contains a. This provides condition $(\mathrm{IPQ})_n$ for the remaining instance $\mathcal{I}'$.

The algorithm runs in polynomial time. Since D is fixed, it is clear that Steps 0–2 can be performed in polynomial time. For Steps 3–5, we also need that m_ℓ is bounded by a polynomial in m, which holds because $\alpha \geq 1/m^2$.

The correctness of the algorithm is given by Theorem 5.1 and the two following propositions, whose proofs can be found in section 8. These propositions show that our new consistency notion satisfies the requirements of step 2 of the general scheme for designing robust approximation algorithms discussed in the introduction.

PROPOSITION 5.2. *The expected total weight of constraints removed by the algorithm is* $O(\alpha^\kappa)$.

PROPOSITION 5.3. *The instance* $\mathcal{I}'$ *obtained after Steps* 0–5 *satisfies the condition* $(\mathrm{IPQ})_n$ *(with the sets* D_x^ℓ *computed by preprocessing step* 2 *in section* 5.3.1*).*

6. Overview of the proof of Theorem 3.1(2). Since the dual discriminator is a majority operation, every relation in Γ is 2-decomposable. Therefore, it follows, e.g., from Lemma 3.2 in [23], that to prove that $\mathrm{CSP}(\Gamma)$ admits a robust algorithm with loss $O(\sqrt{\varepsilon})$, it suffices to prove this for the case when Γ consists of all unary and binary relations preserved by the dual discriminator. Such binary constraints are of one of the four kinds described in section 2.2. Using this description, it follows from Lemma 3.2 of [23] that it suffices to consider the following three types of constraints:

1. disjunction constraints of the form $x = a \vee y = b$, where $a, b \in D$;
2. unique game (UG) constraints of the form $x = \pi(y)$, where π is any permutation on D;
3. unary constraints of the form $x \in P$, where P is an arbitrary nonempty subset of D.

We present an algorithm that, given a $(1-\varepsilon)$-satisfiable instance $\mathcal{I} = (V, \mathcal{C})$ of the problem, finds a solution satisfying constraints with expected total weight $1 - O(\sqrt{\varepsilon \log |D|})$ (the hidden constant in the O-notation depends neither on ε nor on $|D|$).

We now give an informal and somewhat imprecise sketch of the algorithm and its analysis. We present details in section 9. We use the SDP relaxation from section 4. Let us call the value $\|\mathbf{x}_a\|^2$ the SDP weight of the value a for variable x.

Variable partitioning step. The algorithm first solves the SDP relaxation. Then it partitions all variables into three groups $\mathcal{V}_0$, $\mathcal{V}_1$, and $\mathcal{V}_2$ using a threshold rounding algorithm with a random threshold. If most of the SDP weight for x is concentrated on one value $a \in D$, then the algorithm puts x in the set $\mathcal{V}_0$ and assigns x the value a. If most of the SDP weight for x is concentrated on two values $a, b \in D$, then the algorithm puts x in the set $\mathcal{V}_1$ and restricts the domain of x to the set $D_x = \{a, b\}$ (thus we guarantee that the algorithm will eventually assign one of the values a or b to x). Finally, if the SDP weight for x is spread among three or more values, then we put x in the set $\mathcal{V}_2$; we do not restrict the domain for such x. After we assign values to $x \in \mathcal{V}_0$ and restrict the domain of $x \in \mathcal{V}_1$ to D_x, some constraints are guaranteed to be satisfied (say, the constraint $(x = a) \vee (y = b)$ is satisfied if we assign x the value a and the constraint $x \in P$ is satisfied if $D_x \subseteq P$). Denote the set

of such constraints by $\mathcal{C}_s$, and let $\mathcal{C}' = \mathcal{C} \setminus \mathcal{C}_s$.

We then identify a set $\mathcal{C}_v \subseteq \mathcal{C}'$ of constraints that we conservatively label as violated. This set includes all constraints in $\mathcal{C}'$, except those belonging to one of the following four groups:

1. disjunction constraints $(x = a) \vee (y = b)$ with $x, y \in \mathcal{V}_1$ and $a \in D_x$, $b \in D_y$;
2. UG constraints $x = \pi(y)$ with $x, y \in \mathcal{V}_1$ and $D_x = \pi(D_y)$;
3. UG constraints $x = \pi(y)$ with $x, y \in \mathcal{V}_2$;
4. unary constraints $x \in P$ with $x \in \mathcal{V}_2$.

Our construction of sets $\mathcal{V}_0$, $\mathcal{V}_1$, and $\mathcal{V}_2$, which is based on randomized threshold rounding, ensures that the expected total weight of constraints in $\mathcal{C}_v$ is $O(\varepsilon)$ (see Lemma 9.2).

The constraints from the four groups above naturally form two disjoint subinstances of $\mathcal{I}$: $\mathcal{I}_1$ (groups 1 and 2) on the set of variables $\mathcal{V}_1$ and $\mathcal{I}_2$ (groups 3 and 4) on $\mathcal{V}_2$. We treat these instances independently, as described below.

Solving instance $\mathcal{I}_1$. The instance $\mathcal{I}_1$ with the domain of each x restricted to D_x is effectively an instance of Boolean 2-CSP (i.e., each variable has a 2-element domain and all constraints are binary). A robust algorithm with quadratic loss for this problem was given by Charikar, Makarychev, and Makarychev [17]. This algorithm finds a solution violating an $O(\sqrt{\varepsilon})$ fraction of all constraints if the optimal solution violates at most an ε fraction of all constraints or $\mathrm{SDPOpt}(\mathcal{I}_1) \leq \varepsilon$. However, we cannot apply this algorithm to the instance $\mathcal{I}_1$ as is. The problem is that the weight of violated constraints in the optimal solution for $\mathcal{I}_1$ may be greater than $\omega(\varepsilon)$. Note that the unknown optimal solution for the original instance $\mathcal{I}$ may assign values to variables x outside of the restricted domain D_x, and hence it is not a feasible solution for $\mathcal{I}_1$. Furthermore, we do not have a feasible SDP solution for the instance $\mathcal{I}_1$ since the original SDP solution (restricted to the variables in $\mathcal{V}_1$) is not a feasible solution for the Boolean 2-CSP problem (because $\sum_{a \in D_x} \mathbf{x}_a$ is not necessarily equal to $\mathbf{v}_0$ and, consequently, $\sum_{a \in D_x} \|\mathbf{x}_a\|^2$ may be less than 1). Thus, our algorithm first transforms the SDP solution to obtain a feasible solution for $\mathcal{I}_1$. To this end, it partitions the set of vectors $\{\mathbf{x}_a : x \in \mathcal{V}_1, a \in D_x\}$ into two sets H and $\bar{H}$ using a modification of the hyperplane rounding algorithm by Goemans and Williamson [26]. In this partitioning, for every variable x, one of the two vectors $\{\mathbf{x}_a : a \in D_x\}$ belongs to H and the other belongs to $\bar{H}$. Label the elements of each D_x as α_x and β_x so that $\mathbf{x}_{\alpha_x}$ is the vector in H and $\mathbf{x}_{\beta_x}$ is the vector in $\bar{H}$. For every x, we define two new vectors $\tilde{\mathbf{x}}_{\alpha_x} = \mathbf{x}_{\alpha_x}$ and $\tilde{\mathbf{x}}_{\beta_x} = \mathbf{v}_0 - \mathbf{x}_{\alpha_x}$. It is not hard to verify that the set of vectors $\{\tilde{\mathbf{x}}_a : x \in \mathcal{V}_1, a \in D_x\}$ forms a feasible SDP solution for the instance $\mathcal{I}_1$. We show that for each disjunction constraint C in the instance $\mathcal{I}_1$, the cost of C in the new SDP solution is not greater than the cost of C in the original SDP solution (see Lemma 9.4). The same is true for all but an $O(\sqrt{\varepsilon})$ fraction of UG constraints. Thus, after removing UG constraints for which the SDP value has increased, we get an SDP solution of cost $O(\varepsilon)$. Using the algorithm [17] for Boolean 2-CSP, we obtain a solution for $\mathcal{I}_1$ that violates constraints of total weight at most $O(\sqrt{\varepsilon})$.

Solving instance $\mathcal{I}_2$. The instance $\mathcal{I}_2$ may contain only unary and UG constraints, as all disjunction constraints are removed from $\mathcal{I}_2$ in the variable partitioning step. We run the approximation algorithm for Unique Games by Charikar, Makarychev, and Makarychev [16] on $\mathcal{I}_2$ using the original SDP solution restricted to vectors $\{\mathbf{x}_a : x \in \mathcal{V}_2, a \in D\}$. This is a valid SDP relaxation because in the instance $\mathcal{I}_2$, unlike the instance $\mathcal{I}_1$, we do not restrict the domain of variables x to D_x. The cost of this SDP solution is at most ε. As shown in [16], the weight of constraints

violated by the algorithm [16] is at most $O(\sqrt{\varepsilon \log |D|})$.

We get the solution for $\mathcal{I}$ by combining solutions for $\mathcal{I}_1$ and $\mathcal{I}_2$ and assigning values chosen at the variable partitioning step to the variables from the set $\mathcal{V}_0$.

7. Proof of Theorem 5.1. In this section, we prove Theorem 5.1. The proof will use constraint languages with relations of arity greater than two. In order to talk about such instances, we need to extend the definition of a pattern. Note that patterns (in the sense of section 5.1) are instances (with some added structure) and the realizations of patterns are solutions. We use the pattern/instance and solution/realization duality to generalize the notion of a pattern. Moreover, we often treat patterns as instances and (whenever it makes sense) instances as patterns.

We will often talk about path/tree instances; they are defined using the *incidence multigraph*. The incidence multigraph of an instance $\mathcal{J}$ is bipartite, its vertex set consists of variables and constraints of $\mathcal{J}$ (which form the two parts), and if a variable x appears j times in a constraint C, then the vertices corresponding to x and C have j edges between them.

An instance is *connected* if its incidence multigraph is connected; an instance is a *tree instance* if it is connected and its incidence multigraph has no multiple edges and no cycles. A *leaf variable* in a tree instance is a variable which corresponds to a leaf in the incidence multigraph, and we say that two variables are *neighbors* if they appear together in a scope of some constraint (i.e., the corresponding vertices are connected by a path of length 2 in the incidence multigraph). Note that the incidence multigraph of a path pattern in a binary instance (treated as an instance, as described in the first paragraph of this section) is a path and that of an n-tree pattern is a tree with n leaves.

The next definition captures, among other things, the connection between the pattern (treated as an instance) and the instance in which the pattern is defined. Let $\mathcal{J}_1$ and $\mathcal{J}_2$ be two instances over the same constraint language. An *(instance) homomorphism* $e\colon \mathcal{J}_1 \to \mathcal{J}_2$ is a mapping that maps each variable of $\mathcal{J}_1$ to a variable of $\mathcal{J}_2$ and each constraint of $\mathcal{J}_1$ to a constraint of $\mathcal{J}_2$ in such a way that every constraint $((y_1, \dots, y_k), R)$ in $\mathcal{J}_1$ is mapped to $((e(y_1), \dots, e(y_k)), R)$.

Using these new notions, a path pattern in an instance $\mathcal{I}$ (see the definition in section 5.1) can alternatively be defined as an instance, with its beginning and end chosen among the leaf variables, whose incidence graph is a path from beginning to end, together with a homomorphism into $\mathcal{I}$. Similarly, we define a *path pattern* in a (not necessarily binary) instance $\mathcal{I}$ as an instance $\mathcal{J}$, with chosen beginning/end leaf variables, whose incidence graph, after removing all the other vertices of degree one, is a path from beginning to end, together with a homomorphism $e\colon \mathcal{J} \to \mathcal{I}$. The addition of path patterns and propagation are defined analogously to the addition of patterns with binary constraints (see section 5.1).

For a k-ary relation R, let $\mathrm{pr}_i(R) = \{a_i \mid (a_1, \dots, a_i, \dots, a_k) \in R\}$. A CSP instance $\mathcal{J}$ is called *arc-consistent* in sets D_x (x ranges over variables of $\mathcal{J}$) if, for any variable x and any constraint $((x_1, \dots, x_k), R)$ in $\mathcal{J}$, if $x_i = x$, then $\mathrm{pr}_i(R) = D_x$. We say that a CSP instance $\mathcal{J}$ satisfies condition (PQ) in sets D_x if

1. $\mathcal{J}$ is arc-consistent in these sets and
2. for any variable x, any path patterns p, q from x to x, and any $a \in D_x$ there exists j such that $a \in \{a\} + (j(p+q) + p)$.

Note that if the instance $\mathcal{J}$ is binary, then (PQ) implies $(\mathrm{IPQ})_n$ for all n (setting $D^i_x = D$ if $i = |D| + 1$ and $D^i_x = D_x$ if $i < |D| + 1$).

The following fact, a special case of Theorem A.2 in [40], provides solutions for

(PQ) instances.

THEOREM 7.1. *If Γ' is a constraint language with an NU polymorphism, then every instance of* $\mathrm{CSP}(\Gamma')$ *satisfying condition* (PQ) *is satisfiable.*

Finally, a standard algebraic notion has not been defined yet: having fixed Γ over a set D, a subset $A \subseteq D$ is a *subuniverse* if, for any polymorphism g of Γ, we have $g(a_1, a_2, \ldots) \in A$ whenever $a_1, a_2, \ldots \in A$. For any $S \subseteq D$, the subuniverse *generated by* S is defined as

$$\{g(a_1, \ldots, a_r) \mid r \geq 1, a_1, \ldots, a_r \in S, g \text{ is an } r\text{-ary polymorphism of } \Gamma\}.$$

7.1. Into the proof. We begin the proof of Theorem 5.1. We fix a binary language Γ compatible with an $(n+1)$-ary NU polymorphism and an instance $\mathcal{I}$ of $\mathrm{CSP}(\Gamma)$ which satisfies $(\mathrm{IPQ})_n$ with sets D_x^ℓ. Note that we can assume that all D_x^ℓ's are subuniverses. If this is not the case, we replace each D_x^ℓ with the subuniverse generated by it. It is easy to check that (after the change) the instance $\mathcal{I}$ still satisfies $(\mathrm{IPQ})_n$ with such enlarged D_x^ℓ's.

For each variable x, choose and fix an arbitrary index i such that $D_x^i = D_x^{i+1}$ and call it the *level* of x. Note that each variable has a level (since the sets D_x^ℓ are nonempty and ℓ ranges from 1 to $|D|+1$). Let V^i denote the set of variables of level i and $V^{<i}, V^{\leq i}, \ldots$ be defined in the natural way.

Our proof of Theorem 5.1 will proceed by applying Theorem 7.1 to $\mathcal{I}$ restricted to V^1, then to V^2, and so on. However, in order to obtain compatible solutions, we will add constraints to the restricted instances.

7.2. The instances in levels. Let $\mathcal{I}^i$ (for $i \leq |D|$) be the instance defined as follows:

1. V^i is the set of variables of $\mathcal{I}^i$.
2. $\mathcal{I}^i$ contains, for every n-tree pattern t of $\mathcal{I}$, the constraint $((x_1, \ldots, x_k), R)$ defined in the following way: let $v_1, \ldots, v_k$ be all the vertices of t labeled by variables from V^i; then $x_1, \ldots, x_k$ are the labels of $v_1, \ldots, v_k$, respectively, and

$$R = \{(r(v_1), \ldots, r(v_k)) \mid r \text{ is an } i\text{-realization of } t \text{ (i.e., inside sets } D_x^{i+1})\}.$$

This definition has a number of immediate consequences: First, every binary constraint between two variables from V^i is present in $\mathcal{I}^i$ (as it defines a two-element n-tree). Second, note that if some n-tree contains a vertex v_j in V^i which is not a leaf, then by splitting the tree t at v_j (with v_j included in both parts) we obtain two trees defining constraints which together are equivalent to the constraint defined by t. This implies that by including only the constraints defined by n-trees t such that only the leaves can be from V^i, we obtain an equivalent (i.e., having the same set of solutions) instance. Throughout most of the proof, we will be working with such a restricted instance. In this instance, the arity of constraints is bounded by n.

Since the arity of a constraint in $\mathcal{I}^i$ is bounded and the size of the universe is fixed, $\mathcal{I}^i$ is a finite instance, even though some constraints in it can be defined via infinitely many n-tree patterns. It is easy to see that all the relations in the constraints are preserved by all the polymorphisms of Γ.

The instance $\mathcal{I}^i$ is arc-consistent with sets $D_x^i (= D_x^{i+1})$: Let $((x_1, \ldots, x_k), R)$ be a constraint defined by $v_1, \ldots, v_k$ in t, and let $a \in D_{x_j}^i$. By $(\mathrm{IPQ})_n$, there is a realization of t in D_x^{i+1} mapping v_j to a, and thus $D_{x_j}^i \subseteq \mathrm{pr}_j R$. On the other hand,

as $D^i_{x_j} = D^{i+1}_{x_j}$ and every tuple in R comes from a realization inside the sets D^{i+1}_x, we get $\mathrm{pr}_j R \subseteq D^i_{x_j}$.

Next, we show that $\mathcal{I}^i$ has property (PQ). Part 1 of the definition was established in the paragraph above. For part 2, let p and q be arbitrary path patterns from x to x in $\mathcal{I}^i$. Define p' and q' to be the paths of trees in $\mathcal{I}$ obtained, from p and q, respectively, by replacing (in the natural way) each constraint in p and q with the tree that defines it (we use the fact that each constraint is defined by leaves of a tree). We apply property $(\mathrm{IPQ})_n$ for $\mathcal{I}$ with $\ell = i$ and patterns p' and q' to get that, for any $x \in V^i$ and any $a \in D^i_x$, there is a number j such that $a \in \{a\} +^i (j(p' + q') + p')$. The property (PQ) follows immediately.

Since $\mathcal{I}^i$ has the property (PQ), then, by Theorem 7.1, it has a solution. The solution to $\mathcal{I}$ will be obtained by taking the union of appropriately chosen solutions to $\mathcal{I}^1, \dots, \mathcal{I}^{|D|}$.

7.3. Invariant of the iterative construction. A global solution, denoted by $\mathsf{sol}\colon V \to D$, is constructed in steps. At the start, we define it for the variables in V^1 by choosing an arbitrary solution to $\mathcal{I}^1$.

In step i, we extend the definition of sol from $V^{<i}$ to $V^{\le i}$, using a carefully chosen solution to $\mathcal{I}^i$. Our construction will maintain the following condition:

(E_i) every n-tree pattern in $\mathcal{I}$ has a realization inside the sets D^{i+1}_x which agrees with sol on $V^{\le i}$.

Note that, after the first step, the condition (E_1) is guaranteed by the constraints of $\mathcal{I}^1$.

Assume that we are in step i: we have already defined sol on $V^{<i}$, and condition (E_{i-1}) holds. Our goal is to extend sol by a solution of $\mathcal{I}^i$ in such a way that (E_i) holds. The remainder of section 7 is devoted to proving that such a solution exists.

Once we accomplish that, we are done with the proof: Condition (E_i) implies that sol is defined on $V^{\le i}$, and for every constraint $((x, y), R)$ between $x, y \in V^{\le i}$ the pattern from x to y containing a single edge labeled by $((x, y), R)$ is an n-tree. This implies that sol satisfies $((x, y), R)$, i.e., it is a solution on $V^{\le i}$. After establishing ($E_{|D|}$), we obtain a solution to $\mathcal{I}$.

7.4. Restricting $\mathcal{I}^i$. We begin by defining a new instance $\mathcal{K}^i$: it is defined almost identically to $\mathcal{I}^i$, but in part 2 of the definition we require that the realization r sends vertices from $V^{<i}$ according to sol. As in the case of $\mathcal{I}^i$, we can assume that all the constraints are defined by leaves of the tree. Thus, every n-tree pattern with no internal vertices in V^i defines one constraint in $\mathcal{I}^i$ and another in $\mathcal{K}^i$. Just like $\mathcal{I}^i$, the instance $\mathcal{K}^i$ is finite.

Note that we still need to establish that constraints of $\mathcal{K}^i$ are nonempty, but the following claim, where f is the fixed $(n+1)$-ary NU polymorphism, holds independently.

Claim 7.2. *Let $((x_1, \dots, x_k), R)$ and $((x_1, \dots, x_k), R')$ be constraints defined by the same tree t in $\mathcal{I}^i$ and $\mathcal{K}^i$ (respectively). If $\overline{a}_1, \dots, \overline{a}_{n+1} \in R'$, $\overline{a} \in R$, and $j \in \{1, \dots, n+1\}$, then $f(\overline{a}_1, \dots, \overline{a}_{j-1}, \overline{a}, \overline{a}_{j+1}, \dots, \overline{a}_{n+1})$ belongs to R'.*

Proof. Let r_i be a realization of t defining $\overline{a}^i$; this realization sends all the vertices of t labeled by variables from $V^{<i}$ according to sol. Let r be a realization of t defining $\overline{a}$.

Define a function, from vertices of t into D, sending a vertex v to

$$f(r_1(v), \dots, r(v), \dots, r_{n+1}(v))$$

(where $r(v)$ is in position j). This is clearly a realization, and if v is labeled by $x \in V^{<i}$, it sends v according to sol (since f is an NU operation). The new realization witnesses that $f(\overline{a}_1, \dots, \overline{a}_{j-1}, \overline{a}, \overline{a}_{j+1}, \dots, \overline{a}_{n+1})$ belongs to R'. ∎

In order to proceed, we need to show that the instance $\mathcal{K}^i$ contains a nonempty, arc-consistent subinstance, i.e., an arc-consistent instance (in some nonempty sets D_x) obtained from $\mathcal{K}^i$ by restricting every constraint in it so that each coordinate can take value only in the appropriate set D_x.

A proof of this claim is the subject of the next section.

7.5. Arc-consistent subinstance of $\mathcal{K}^i$. In order to proceed with the proof, we need an additional definition. Let $e\colon \mathcal{J}_1 \to \mathcal{J}_2$ be an instance homomorphism. If for any variable y of $\mathcal{J}_1$ and any constraint $((x_1, \dots, x_k), R)$ of $\mathcal{J}_2$ with $e(y) = x_i$ (for some i) the constraint $((x_1, \dots, x_k), R)$ has exactly one preimage $((y_1, \dots, y_k), R)$ with $y = y_i$, we say that e is a *covering*. A *universal covering tree instance* $\mathrm{UCT}(\mathcal{J})$ of a connected instance $\mathcal{J}$ is a (possibly countably infinite) tree instance $\mathcal{T}$ together with a covering $e\colon \mathcal{T} \to \mathcal{J}$ satisfying some additional properties. If $\mathcal{J}$ is a tree instance, then one can take $\mathrm{UCT}(\mathcal{J}) = \mathcal{J}$; otherwise, $\mathrm{UCT}(\mathcal{J})$ is always infinite. If an instance $\mathcal{J}$ is disconnected, then $\mathrm{UCT}(\mathcal{J})$ is a disjoint union of universal covering tree instances for connected components of $\mathcal{J}$.

Several equivalent (precise) definitions of UCT can be found in section 5.4 of [40] or section 4 of [43]. For our purposes, it is enough to mention that, for any $\mathcal{J}$, the instance $\mathrm{UCT}(\mathcal{J})$ (with covering e) has the following two properties. For any two variables v, v' satisfying $e(v) = e(v')$, there exists an endomorphism h of $\mathrm{UCT}(\mathcal{J})$ (i.e., a homomorphism into itself) sending v to v' and such that $e \circ h = e$. Similarly for constraints C and C', if $e(C) = e(C')$, then there is an endomorphism h such that $h(C) = C'$ and $e \circ h = e$. It is well known that $\mathrm{UCT}(\mathcal{J})$ has a solution if and only if $\mathcal{J}$ has an arc-consistent subinstance.

Consider $\mathrm{UCT}(\mathcal{K}^i)$, and fix a covering $e'\colon \mathrm{UCT}(\mathcal{K}^i) \to \mathcal{K}^i$. Let $\mathcal{T}^i$ be an instance obtained from $\mathrm{UCT}(\mathcal{K}^i)$ by replacing each constraint C in it by a tree that defines $e'(C)$, each time introducing a fresh set of variables for the internal vertices of the trees. Let e be the instance homomorphism from $\mathcal{T}^i$ to $\mathcal{I}$ defined in the natural way. We call a solution (or a partial solution) to $\mathcal{T}^i$ *nice* if it maps each v into $D^{i+1}_{e(v)}$, and, moreover, if $e(v) \in V^{<i}$, then v is mapped to $\mathsf{sol}(e(v))$. It should be clear that nice solutions to $\mathcal{T}^i$ correspond to solutions of $\mathrm{UCT}(\mathcal{K}^i)$ (although the correspondence is not one-to-one).

Claim 7.3. *There exists a nice solution of $\mathcal{T}^i$.*

Proof. If $\mathcal{T}^i$ is not connected, we consider each connected component separately and then take the union of nice solutions. Henceforth we assume that $\mathcal{T}^i$ is connected. By a standard compactness argument, it suffices to find a nice solution for every finite subtree of $\mathcal{T}^i$. Suppose, for a contradiction, that $\mathcal{T}$ is a minimal finite subtree of $\mathcal{T}^i$ without nice solutions.

First, only the leaf vertices of $\mathcal{T}$ can be mapped, by e, into variables from $V^{<i}$. Indeed, if an internal vertex is mapped to a variable in $V^{<i}$, we can split the tree at this vertex into two parts, obtain (from the minimality of $\mathcal{T}$) nice solutions to both parts (which need to map the splitting vertex according to sol, i.e., to the same element), and merge these solutions to obtain a nice solution to $\mathcal{T}$. This is a contradiction.

Second, we show that $\mathcal{T}$ has more than n leaves mapped by e into $V^{<i}$. Assume that $\mathcal{T}$ has n or fewer leaves mapped to $V^{<i}$, and let $\mathcal{T}'$ be the smallest subtree of $\mathcal{T}$

with these leaves. Then $\mathcal{T}'$ is an n-tree and by (E_{i-1}) we obtain a solution s to $\mathcal{T}'$ in D^i_x's which sends leaves of $\mathcal{T}'$ according to sol. It remains to extend s to a solution of $\mathcal{T}$ in D^{i+1}_x's. This extension is done in a sequence of steps. In each step, s is defined for increasingly larger subtrees of $\mathcal{T}$. Furthermore, in each step the following condition (*) is satisfied by s: if a vertex v has a value assigned by s and a neighbor without such a value, then $s(v)$ belongs to $D^i_{e(v)}$. Clearly, this condition holds in the beginning. In each step, we pick a constraint C on a vertex v with an assigned value and a vertex v' without such a value. (Note that the constraints of $\mathcal{T}^i$, and consequently of $\mathcal{T}$, are binary.) C has been added to $\mathcal{T}^i$ by replacing a constraint of $\mathrm{UCT}(\mathcal{K}^i)$ with an n-tree $\mathcal{T}_C$ that defines it. Let $\mathcal{S}$ be a maximal subtree of $\mathcal{T}$ such that it contains C, it has v as a leaf, and all other nodes in $\mathcal{S}$ have not been assigned by s and belong to $\mathcal{T}_C$. Since $\mathcal{T}_C$ is an n-tree, $\mathcal{S}$ is also an n-tree, and we can use $(\mathrm{IPQ})_n$ to derive that there exists a solution, s', of $\mathcal{S}$ in D^{i+1}_x's that sends v to $s(v) \in D^i_{e(v)}$. More specifically, we apply $(\mathrm{IPQ})_n$ with $x = v$, $a = s(v)$, and both p and q are the same pattern $t_1 + t_2$ such that t_1 is $\mathcal{S}$ with the beginning v and the end being any other leaf of $\mathcal{S}$, and t_2 is t_1 with the beginning and end swapped. This solution s' can be added to s (as the values on v are the same). It remains to see that condition (*) is preserved after extending s with s'. Indeed, let u be any vertex such that after adding solution s' has a neighbor u' that has not yet been assigned. We can assume that u is one of the new variables assigned by s'. If $e(u) \in V^i$, then the claim follows from the fact that $D^{i+1}_{e(u)} = D^i_{e(u)}$, and so we can assume that $e(u) \notin V^i$. However, in this case, all neighbors of u in $\mathcal{T}$ must be in $\mathcal{T}_C$, so the constraint in $\mathcal{T}$ containing both u and u' must be also in $\mathcal{T}_C$, contradicting the maximality of $\mathcal{S}$.

So the counterexample $\mathcal{T}$ must have at least $n+1$ leaves mapped into $V^{<i}$. Fix any $n+1$ of such leaves $v_1, \dots, v_{n+1}$, and let $\mathcal{T}_j$, for $j = 1, \dots, n+1$, denote a subinstance of $\mathcal{T}$ obtained by removing v_j together with the single constraint containing v_j: $((v_j, v'_j), R_j)$ from $\mathcal{T}$. Clearly, v'_j is not a leaf (as it would make our $\mathcal{T}$ a two-element instance), and by the fact that only leaves can be mapped into $V^{<i}$ we get that $e(v'_j) \in V^i$ or $e(v'_j) \in V^{>i}$ and, in the last case, $i \neq |D|$.

By minimality, each $\mathcal{T}_j$ has a nice realization, say s_j. Now either $e(v'_j) \in V^i$ and $s_j(v'_j) \in D^i_{e(v'_j)} = D^{i+1}_{e(v'_j)}$ or $e(v'_j) \in V^{>i}, s_i(v'_j) \in D^{i+1}_{e(v'_j)}$ and $i+1 \neq |D|+1$. In both cases, $s_j(v'_j) \in D^{i'}_{e(v'_j)}$ for $i' \leq |D|$, and thus, by $(\mathrm{IPQ})_n$, there exists $a_j \in D$ such that $(a_j, s_j(v'_j)) \in R_i$. We let s'_j be the realization of $\mathcal{T}$ obtained by extending s_j by mapping v_j to a_j. The last step is to apply the $(n+1)$-ary NU operation coordinatewise to s'_j's (in a way identical to the one in the proof of Claim 7.2). The application produces a nice realization of $\mathcal{T}$. This contradiction finishes the proof of the claim. ∎

We will denote the arc-consistent subinstance of $\mathcal{K}^i$ (which is about to be constructed) by $\mathcal{L}^i$. The variables of $\mathcal{L}^i$ and $\mathcal{K}^i$ (or indeed $\mathcal{I}^i$) are the same. For every constraint $(\overline{x}, R)$ in $\mathcal{K}^i$, we introduce a constraint $(\overline{x}, R')$ into $\mathcal{L}^i$, where

$$R' = \{\overline{a} \colon \overline{a} = s(\overline{y}), \text{ where } s \text{ is a solution to } \mathrm{UCT}(\mathcal{K}^i) \text{ and } e'((\overline{y}, R)) = (\overline{x}, R)\},$$

where e' is an instance homomorphism mapping $\mathrm{UCT}(\mathcal{K}^i)$ to $\mathcal{K}^i$. In other words, we restrict a relation in a constraint of $\mathcal{K}^i$ by allowing only the tuples which appear in a solution of the $\mathrm{UCT}(\mathcal{K}^i)$ (at this constraint).

All the relations of $\mathcal{L}^i$ are preserved by all the polymorphisms of Γ and are nonempty (by Claim 7.2). The fact that $\mathcal{L}^i$ is arc-consistent is an easy consequence of the endomorphism structure of universal covering trees. Finally, Claim 7.2 holds

for $\mathcal{L}^i$.

CLAIM 7.4. *Let $((x_1,\dots,x_k),R)$ and $((x_1,\dots,x_k),R')$ be constraints defined by the same tree t in $\mathcal{I}^i$ and $\mathcal{L}^i$, respectively. Let $\overline{a}_1,\dots,\overline{a}_{n+1}\in R'$ and $\overline{a}\in R$; then $f(\overline{a}_1,\dots,\overline{a},\dots,\overline{a}_{n+1})$, where f is the $(n+1)$-ary NU operation and $\overline{a}$ is in position j, belongs to R'.*

Proof. By Claim 7.2, the tuple $f(\overline{a}_1,\dots,\overline{a},\dots,\overline{a}_{n+1})$ belongs to the relation in the corresponding constraint in $\mathcal{K}^i$. Thus, if it extends to a solution of $\mathrm{UCT}(\mathcal{K}^i)$, it belongs to R'. However, each $\overline{a}^i$ extends to a solution of $\mathrm{UCT}(\mathcal{K}^i)$ and $\overline{a}$ extends to a solution of $\mathrm{UCT}(\mathcal{I}^i)$. By applying the NU operation f to these extensions (coordinatewise), we obtain the required evaluation. ∎

7.6. A solution to $\mathcal{K}^i$. In order to find a solution to $\mathcal{L}^i$, we will use Corollary B.2 from [40]. We state it here in a simplified form using the following notation: for subuniverses $A'\subseteq A$, we say that A' *nu-absorbs* A if, for some NU polymorphism f, $f(a_1,\dots,a_n)\in A'$ whenever $a_1,\dots,a_n\in A$ and at most one a_i is in $A\setminus A'$. Similarly, if $R'\subseteq R$ are relations preserved by all polymorphisms of Γ, we say R' nu-absorbs R if for some NU operation f taking all arguments from R' except for one which comes from R produces a result in R'.

COROLLARY 7.5 (Corollary B.2 from [40]). *Let $\mathcal{I}$ satisfy the* (PQ) *condition in sets A_x. Let $\mathcal{I}'$ be an arc-consistent instance in sets A'_x on the same set of variables as $\mathcal{I}$ such that the following hold:*

1. *for every variable x, the subuniverse A'_x nu-absorbs A_x; and*
2. *for every constraint $((x_1,\dots,x_n),R')$ in $\mathcal{I}'$, there is a corresponding constraint $((x_1,\dots,x_n),R)$ in $\mathcal{I}$ such that R' nu-absorbs R (and both respect the NU operation).*

Then there are subuniverses A''_x of A'_x (for every x) such that the instance $\mathcal{I}''$ obtained from $\mathcal{I}'$ by restricting the domain of each variable to A''_x and by restricting the constraint relations accordingly satisfies the condition (PQ).

We will apply the corollary above using $\mathcal{I}^i$ for $\mathcal{I}$ and $\mathcal{L}^i$ for $\mathcal{I}'$. By our construction, $\mathcal{I}^i$ satisfies condition (PQ), and the sets D^i_x (which play the role of A_x) are subuniverses of D. On the other hand, $\mathcal{L}^i$ is arc-consistent and all the relations involved in it are closed under the polymorphisms of Γ. Claim 7.4 shows that each relation R' nu-absorbs the corresponding R. By arc-consistency, the projection of R' on a variable x is the same for each constraint $((x_1,\dots,x_n),R')$ containing x; call the corresponding sets A'_x. Since each R' nu-absorbs R, it follows that each A'_x nu-absorbs the corresponding A_x. The corollary implies that we can restrict the instance $\mathcal{L}^i$ to obtain an instance satisfying (PQ). By Theorem 7.1, such an instance, and thus both $\mathcal{K}^i$ and $\mathcal{L}^i$, has a solution.

7.7. Finishing the proof. We choose any solution to $\mathcal{K}^i$ and extend the global solution sol to V^i according to it. There exists a solution on $V^{\le i}$ because every constraint between two variables from this set is either in $V^{<i}$ or defines a two-variable n-tree which was used to define a constraint in $\mathcal{K}^i$. It remains to prove that, with such an extension, condition (E_i) holds.

Let t be an n-tree pattern in $\mathcal{I}$. If it has no variables mapped to V^i, then (E_i) follows from (E_{i-1}). Assume that it has such variables. By splitting t at internal vertices mapped to V^i, it is enough to consider the case when only leaves of t are mapped to V^i. Then t defines a constraint $(\overline{x},R)$ in $\mathcal{K}^i$. The solution to $\mathcal{K}^i$ mapping $\overline{x}$ to $\overline{a}\in R$ and the evaluation of t witnessing that $\overline{a}$ belongs to R can be taken to

satisfy (E_i) for t. Theorem 5.1 is proved.

8. Full proof of Theorem 3.1(1). In this subsection, we prove Propositions 5.2 and 5.3. The following equalities, which can be directly verified, are used repeatedly in this section: for any subsets A, B of D and any feasible solution $\{\mathbf{x}_a\}$ of the SDP relaxation of $\mathcal{I}$, it holds that $\|\mathbf{x}_A\|^2 = \mathbf{x}_A\mathbf{y}_D$ and $\|\mathbf{y}_B - \mathbf{x}_A\|^2 = \mathbf{x}_{D\setminus A}\mathbf{y}_B + \mathbf{x}_A\mathbf{y}_{D\setminus B}$.

8.1. Analysis of preprocessing step 2. In some of the proofs, it will be required that $\alpha \le c_0$ for some constant c_0 depending only on $|D|$. This can be assumed without loss of generality since we can adjust constants in O-notation in Theorem 3.1(1) to ensure that $\varepsilon \le c_0$ (and we know that $\alpha \le \varepsilon$). We will specify the requirements on the choice of c_0 as we go along.

LEMMA 8.1. *There exists a constant $c > 0$ that depends only on $|D|$ such that the sets $D_x^\ell \subseteq D$, $x \in V$, $1 \le \ell \le |D|$, obtained in preprocessing step 2 are nonempty and satisfy the following conditions:*

(1) *for every $a \in D_x^\ell$, $\|\mathbf{x}_a\| \ge \alpha^{3\ell\kappa}$;*
(2) *for every $a \notin D_x^\ell$, $\|\mathbf{x}_a\| \le c\alpha^{3\ell\kappa}$;*
(3) *for every $a \in D_x^\ell$, $\|\mathbf{x}_a\|^2 \ge 2\|\mathbf{x}_{D\setminus D_x^\ell}\|^2$;*
(4) *$D_x^\ell \subseteq D_x^{\ell+1}$ (with $D_x^{|D|+1} = D$).*

Proof. Let $c = (2|D|)^{(|D|/2)}$. It is straightforward to verify that conditions 1–3 are satisfied. Let us show condition 4. Since c only depends on $|D|$, we can choose c_0 (an upper bound on α) so that $c\alpha^{3\kappa} < 1$. It follows that $c\alpha^{3(\ell+1)\kappa} < \alpha^{3\ell\kappa}$. It follows from conditions 1 and 2 that $D_x^\ell \subseteq D_x^{\ell+1}$.

Finally, let us show that D_x^ℓ is nonempty. By condition 4, we only need to take care of case $\ell = 1$. We have by condition 2 that

$$\sum_{a\in D\setminus D_x^1} \|\mathbf{x}_a\|^2 \le |D|c^2\alpha^{6\kappa}.$$

Note that we can adjust c_0 to also satisfy $|D|c^2\alpha^{6\kappa} < 1$ because, again, c only depends on $|D|$. ∎

8.2. Proof of Proposition 5.2. We will prove that the total weight of constraints removed in Steps 0–5 of the algorithm in section 5.3.2 is $O(\alpha^\kappa)$.

LEMMA 8.2. *The total weight of the constraints removed in Step 0 is at most α^κ.*

Proof. We have

$$\alpha \ge \sum_{C\in\mathcal{C}} w_C \operatorname{loss}(C) \ge \sum_{\substack{C\in\mathcal{C}\\ \operatorname{loss}(C)\ge\alpha^{1-\kappa}}} w_C\alpha^{1-\kappa},$$

from which the lemma follows. ∎

LEMMA 8.3. *Let $((x,y),R)$ be a constraint not removed in Step 0, and let A, B be such that $B = A +^\ell (x,R,y)$. Then $\|\mathbf{y}_B\|^2 \ge \|\mathbf{x}_A\|^2 - c\alpha^{(6\ell+6)\kappa}$ for some constant $c > 0$ depending only on $|D|$. The same is also true for a constraint $((y,x),R)$ and $A = B +^\ell (y,R^{-1},x)$.*

Proof. Consider the first case, i.e., a constraint $((x,y),R)$ and $B = A +^\ell (x,R,y)$. We have

$$\mathbf{x}_A\mathbf{y}_{D\setminus B} = \sum_{\substack{a\in A, b\in D\setminus B\\ (a,b)\notin R}} \mathbf{x}_a\mathbf{y}_b + \sum_{\substack{a\in A, b\in D\setminus B\\ (a,b)\in R}} \mathbf{x}_a\mathbf{y}_b.$$

The first term is bounded from above by the loss of constraint $((x,y),R)$, and hence is at most $\alpha^{1-\kappa}$, since the constraint has not been removed in Step 0. Since $B = A +^{\ell} (x,R,y)$, it follows that for every $(a,b) \in R$ such that $a \in A$ and $b \in D \setminus B$ we have that $a \notin D_x^{\ell+1}$ or $b \notin D_y^{\ell+1}$. Hence, the second term is at most

$$\mathbf{x}_{D \setminus D_x^{\ell+1}} \mathbf{y}_D + \mathbf{x}_D \mathbf{y}_{D \setminus D_y^{\ell+1}} = \|\mathbf{x}_{D \setminus D_x^{\ell+1}}\|^2 + \|\mathbf{y}_{D \setminus D_y^{\ell+1}}\|^2,$$

which, by Lemma 8.1(2), is bounded from above by $d\alpha^{(6\ell+6)\kappa}$ for some constant $d > 0$. From the definition of κ, it follows that $(6\ell+6)\kappa \leq 1-\kappa$, and hence we conclude that $\mathbf{x}_A \mathbf{y}_{D \setminus B} \leq (d+1)\alpha^{(6\ell+6)\kappa}$. Then we have that

$$\begin{aligned} \|\mathbf{y}_B\|^2 &= \mathbf{x}_A \mathbf{y}_B + \mathbf{x}_{D \setminus A} \mathbf{y}_B \geq \mathbf{x}_A \mathbf{y}_B = \mathbf{x}_A \mathbf{y}_D - \mathbf{x}_A \mathbf{y}_{D \setminus B} \\ &\geq \|\mathbf{x}_A\|^2 - (d+1)\alpha^{(6\ell+6)\kappa}. \end{aligned}$$

∎

LEMMA 8.4. *The expected weight of the constraints removed in Step* 1 *is* $O(\alpha^{\kappa})$.

Proof. Let $((x,y),R)$ be a constraint not removed in Step 0. We shall see that the probability that it is removed in Step 1 is at most $c\alpha^{\kappa}$, where $c > 0$ is a constant.

Let A, B be such that $B = A +^{\ell} (x,R,y)$. It follows from Lemma 8.3 that $\|\mathbf{y}_B\|^2 \geq \|\mathbf{x}_A\|^2 - d\alpha^{(6\ell+6)\kappa}$ for some constant $d > 0$. Hence, the probability that a value r_ℓ in Step 1 makes that $\mathbf{y}_B \not\preceq^{\ell} \mathbf{x}_A$ is at most

$$\frac{d\alpha^{(6\ell+6)\kappa}}{\alpha^{(6\ell+4)\kappa}} = d\alpha^{2\kappa} \leq d\alpha^{\kappa}.$$

We obtain the same bound if we switch x and y and consider sets A, B such that $A = B +^{\ell} R^{-1}$. Taking the union bound for all sets A, B and all values of ℓ, we obtain the desired bound. ∎

LEMMA 8.5. *If there exists a constant* $c > 0$ *depending only on* $|D|$ *such that for every variable* x *the probability that all constraints involving* x *are removed in Step* 2*, Step* 3*, or Step* 5 *is at most* $c\alpha^{\kappa}$*, then the total expected weight of constraints removed this way in the corresponding is at most* $2c\alpha^{\kappa}$.

Proof. Let w_x denote the total weight of the constraints in which x participates. The expected weight of constraints removed is at most

$$\sum_{x \in V} w_x c\alpha^{\kappa} = \left(\sum_{x \in V} w_x \right) c\alpha^{\kappa} = 2c\alpha^{\kappa},$$

and the lemma is proved. ∎

LEMMA 8.6. *The expected weight of the constraints removed in Step* 2 *is* $O(\alpha^{\kappa})$.

Proof. Let x be a variable. According to Lemma 8.5, it is enough to prove that the probability that we remove all constraints involving x in Step 2 is at most $c\alpha^{\kappa}$ for some constant $c > 0$. Suppose that $A \subseteq B$ are such that $\|\mathbf{x}_B\|^2 - \|\mathbf{x}_A\|^2 = \|\mathbf{x}_B - \mathbf{x}_A\|^2 \leq (2n-3)\alpha^{(6\ell+4)\kappa}$. Then the probability that one of the bounds of the form $r_\ell + (s_\ell + jm_0)\alpha^{(6\ell+4)\kappa}$ separates $\|\mathbf{x}_B\|^2$ and $\|\mathbf{x}_A\|^2$ is at most

$$(2n-3)/m_0 \leq (2n-3)/(\alpha^{-2\kappa} - 1),$$

which is at most $c\alpha^{\kappa}$ for some constant $c > 0$ whenever $\alpha^{\kappa} < 1/2$. The latter can be ensured by adjusting constant c_0 from section 8.1. Taking the union bound for all sets A, B and all values of ℓ, we obtain the desired bound. ∎

LEMMA 8.7. *There exist constants $c, d > 0$ depending only on $|D|$ such that for every pair of variables x and y and every $A, B \subseteq D$ the probability, p, that a unit vector $\mathbf{u}$ chosen uniformly at random cuts $\mathbf{x}_A$ and $\mathbf{y}_B$ satisfies*

$$c \cdot \|\mathbf{y}_B - \mathbf{x}_A\| \leq p \leq d \cdot \|\mathbf{y}_B - \mathbf{x}_A\|.$$

Proof. Let $0 \leq x \leq 1$, and let $0 \leq \theta \leq \pi$ be an angle such that $x = \cos(\theta)$. There exist constants $a, b > 0$ such that

$$a \cdot \sqrt{1-x} \leq \theta \leq b \cdot \sqrt{1-x}.$$

Now, if θ is the angle between $\mathbf{x}_A - \mathbf{x}_{D\setminus A}$ and $\mathbf{y}_B - \mathbf{y}_{D\setminus B}$, then

$$\begin{aligned} 1 - \cos(\theta) &= 1 - (\mathbf{x}_A - \mathbf{x}_{D\setminus A})(\mathbf{y}_B - \mathbf{y}_{D\setminus B}) = 2(\mathbf{x}_{D\setminus A}\mathbf{y}_B + \mathbf{x}_A\mathbf{y}_{D\setminus B}) \\ &= 2\|\mathbf{y}_B - \mathbf{x}_A\|^2. \end{aligned}$$

Since $p = \theta/\pi$, the result follows. ∎

LEMMA 8.8. *The expected weight of the constraints removed in Step 3 is $O(\alpha^\kappa)$.*

Proof. According to Lemma 8.5, it is enough to prove that the probability that we remove all constraints involving x in Step 3 is at most $c\alpha^\kappa$ for some constant c. Let A and B be such that $A \cap D_x^\ell \neq B \cap D_x^\ell$. Let a be an element in symmetric difference $(A \cap D_x^\ell) \triangle (B \cap D_x^\ell)$. Then we have $\|\mathbf{x}_B - \mathbf{x}_A\| = \sqrt{\mathbf{x}_{D\setminus A}\mathbf{x}_B + \mathbf{x}_A\mathbf{x}_{D\setminus B}} \geq \|\mathbf{x}_a\| \geq \alpha^{3\ell\kappa}$, where the last inequality is by Lemma 8.1(1). Then by Lemma 8.7 the probability that $\mathbf{x}_A$ and $\mathbf{x}_B$ are not ℓ-cut is at most

$$(1 - c\alpha^{3\ell\kappa})^{m_\ell} \leq \frac{1}{\exp(c\alpha^{3\ell\kappa} m_\ell)} \leq \frac{1}{\exp(c\alpha^{-\kappa})} \leq c\alpha^\kappa,$$

where c is the constant given in Lemma 8.7. Taking the union bound for all sets A, B and all values of ℓ, we obtain the desired bound. ∎

LEMMA 8.9. *The expected weight of the constraints removed in Step 4 is $O(\alpha^\kappa)$.*

Proof. Let $((x, y), R)$ be a constraint not removed in Steps 0 and 1. We shall prove that the probability that it is removed in Step 4 is at most $c\alpha^\kappa$ for some constant $c > 0$.

Fix ℓ and A, B such that $B = A +^\ell (x, R, y)$. Since the constraint has not been removed in Step 1, we have $\mathbf{y}_B \preceq^\ell \mathbf{x}_A$. Since $B = A +^\ell p$, we have that $\mathbf{x}_A\mathbf{y}_{D\setminus B} \leq c_1\alpha^{(6\ell+6)\kappa}$, as shown in the proof of Lemma 8.3. Since $\|\mathbf{x}_A\|^2 = \mathbf{x}_A(\mathbf{y}_B + \mathbf{y}_{D\setminus B})$, it follows that $\mathbf{x}_A\mathbf{y}_B \geq \|\mathbf{x}_A\|^2 - c_1\alpha^{(6\ell+6)\kappa}$.

Also, we have that $\|\mathbf{y}_B\|^2 = (\mathbf{x}_A\mathbf{y}_B + \mathbf{x}_{D\setminus A}\mathbf{y}_B)$ is at most $\|\mathbf{x}_A\|^2 + \alpha^{(6\ell+4)\kappa}$ because $\mathbf{y}_B \preceq^\ell \mathbf{x}_A$. Using the bound on $\mathbf{x}_A\mathbf{y}_B$ obtained above, it follows that $\mathbf{x}_{D\setminus A}\mathbf{y}_B$ is at most $\alpha^{(6\ell+4)\kappa} + c_1\alpha^{(6\ell+6)\kappa} \leq (c_1+1)\alpha^{(6\ell+4)\kappa}$.

Putting the bounds together, we have that

$$\begin{aligned} \|\mathbf{y}_B - \mathbf{x}_A\| &= \sqrt{\mathbf{x}_{D\setminus A}\mathbf{y}_B + \mathbf{x}_A\mathbf{y}_{D\setminus B}} \leq \sqrt{c_1\alpha^{(6\ell+6)\kappa} + (c_1+1)\alpha^{(6\ell+4)\kappa}} \\ &\leq c_2\alpha^{(3\ell+2)\kappa} \end{aligned}$$

for some constant $c_2 > 0$.

Applying the union bound and Lemma 8.7, we have that the probability that $\mathbf{x}_A$ and $\mathbf{y}_B$ are ℓ-cut is at most $m_\ell d c_2 \alpha^{(3\ell+2)\kappa} = O(\alpha^\kappa)$. We obtain the same bound if we switch x and y and take R^{-1} instead of R. Taking the union bound for all sets A, B and all values of ℓ, we obtain the desired bound. ∎

LEMMA 8.10. *The expected weight of the constraints removed in step* 5 *is* $O(\alpha^\kappa)$.

Proof. Again, according to Lemma 8.5, it is enough to prove that the probability that we remove all constraints involving x in Step 5 is at most $c_1\alpha^\kappa$ for some constant c_1. Suppose that A, B are such that $\|\mathbf{x}_A - \mathbf{x}_B\|^2 \leq (2n-3)\alpha^{(6\ell+4)\kappa}$. Hence, by Lemma 8.7 and the union bound, the probability that $\mathbf{x}_A$ and $\mathbf{x}_B$ are ℓ-cut is at most

$$m_\ell d(2n-3)^{1/2}\alpha^{(3\ell+2)\kappa} \leq d(2n-3)^{1/2}\alpha^\kappa,$$

where d is the constant from Lemma 8.7. Taking the union bound for all sets A, B and all values of ℓ, we obtain the desired bound. ∎

8.3. Proof of Proposition 5.3. All patterns appearing in this subsection are in $\mathcal{I}'$. The following notion will be used several times in our proofs: Let t be a tree, and let y be one of its nodes. We say that a subtree t' of t is *separated by vertex* y if t' is maximal among all the subtrees of t that contain y as a leaf.

In the first part of the proof (which consists of the following three lemmas), we prove that if we start with a set $A \subseteq D_x$ and propagate it via a path p, from x to y, of n-tree patterns to obtain a set $B \subseteq D_y$, the value $\|\mathbf{y}_B\|$ cannot be much smaller than $\|\mathbf{x}_A\|$. The first lemma proves that this is the case if we restrict ourselves to proper path patterns.

LEMMA 8.11. *Let* $1 \leq \ell \leq |D|$, *let* p *be a path pattern from* x *to* y, *and let* A, B *be such that* $B = A +^\ell p$. *Then* $\mathbf{x}_A \preceq^\ell \mathbf{y}_B$ *and, in particular,* $\|\mathbf{x}_A\| \leq \|\mathbf{y}_B\| + \alpha^{(6\ell+4)\kappa}$.

Proof. Since the relation $\preceq^\ell$ is transitive, it is enough to prove the lemma for path patterns containing only one constraint. But this is true since all the constraints $((x,y),R)$ or $((y,x),R)$ which would invalidate the lemma have been removed in Step 1. ∎

The second lemma proves that the weight of sets that vanish after following a tree pattern is small.

LEMMA 8.12. *If* p *is a tree pattern with at most* $j+1$ *leaves starting at* x, *and* $A \subseteq D_x^{\ell+1}$ *is such that* $A +^\ell p = \emptyset$, *then* $\|\mathbf{x}_A\|^2 \leq (2j-1)\alpha^{(6\ell+4)\kappa}$.

Proof. We will prove the statement by induction on the number of leaves. For $j = 1$, this follows from Lemma 8.11. Suppose, then, that p is a tree pattern with $j+1 > 2$ leaves and the statement is true for any tree pattern with at most j leaves. Choose y to be the first branching vertex in the unique path in p from x to the end of p, and let $p_0, t_1, \ldots, t_h$ be all subtrees of p separated by y, where p_0 is the subtree containing x. We turn p_0 into a pattern by choosing x as the beginning and y as the end. Similarly, we turn every t_i into a pattern by choosing y as the beginning and any other arbitrary leaf as the end. Since y is a branching vertex, we have that $h \geq 2$, every t_i has $j_i + 1 < j+1$ leaves, and $\sum_{i=1}^h j_i = j$. Now let B_i denote the set $\{a \in D_y^{\ell+1} : \{a\} +^\ell t_i = \emptyset\}$. Since $j_i < j$, we know that $\|\mathbf{y}_{B_i}\|^2 \leq (2j_i-1)\alpha^{(6\ell+4)\kappa}$. Further, for $B = \bigcup_{i=1}^h B_i$, we have, using inductive assumption, that

$$\begin{aligned}\|\mathbf{y}_B\|^2 &\leq \sum_{i=1}^h \|\mathbf{y}_{B_i}\|^2 \leq \sum_{i=1}^h (2j_i-1)\alpha^{(6\ell+4)\kappa} = (2j-h)\alpha^{(6\ell+4)\kappa} \\ &\leq (2j-2)\alpha^{(6\ell+4)\kappa}.\end{aligned}$$

Finally, since $A +^\ell p = \emptyset$, then $A +^\ell p_0 \subseteq B$, and the required claim follows from Lemma 8.11. ∎

The following lemma concludes the first part of the proof by proving that following a path of n-trees pattern cannot decrease the weight of a set too much.

LEMMA 8.13. *Let $1 \leq \ell \leq |D|$, and let p be a pattern from x to y which is a path of n-trees. If $A, B \subseteq D$ are such that $A +^{\ell} p = B$, then $\|\mathbf{x}_A\|^2 \leq \|\mathbf{y}_B\|^2 + \alpha^{(6\ell+2)\kappa}$.*

Proof. We claim that for any n-tree pattern t and A, B with $A +^{\ell} t = B$, we have $\mathbf{x}_A \preceq^{\ell}_w \mathbf{y}_B$. Since the relation $\preceq^{\ell}_w$ is transitive, the lemma is then a direct consequence. For a contradiction, suppose that t is a smallest (by inclusion) n-tree that does not satisfy the claim. Observe that t is not a path, due to Lemma 8.11 and the fact that $\mathbf{x}_A \preceq^{\ell} \mathbf{y}_B$ implies $\mathbf{x}_A \preceq^{\ell}_w \mathbf{y}_B$. Let v_x and v_y denote the beginning and the end vertices of t, respectively; and let v_z be the last branching vertex that appears on the path connecting v_x and v_y, and let it be labeled by z. Let $t_1, t_2, p_1, \ldots, p_j$ be all subtrees of t separated by v_z, where t_1 and t_2 are the subtrees containing v_x and v_y, respectively. Let us turn $p_1, \ldots, p_j$ into patterns by choosing v_z as the beginning and any other leaf as the end. Note that the sum of numbers of the leaves of $p_1, \ldots, p_j$ when excluding v_z is less than $n-1$ since t was a path of n-trees. Furthermore, choose x and z to be the beginning and end, respectively, of t_1 and z and y to be the beginning and end, respectively, of t_2. Note that t_2 is a path. Further, we know that for $C = A +^{\ell} t_1$ we have $\mathbf{x}_A \preceq^{\ell}_w \mathbf{z}_C$ by the minimality of t. Now let $C_i = \{a \in D^{\ell+1}_z : \{a\} +^{\ell} p_i = \emptyset\}$. Then, by Lemma 8.12, we get that $\|\mathbf{z}_{C_i}\|^2 \leq (2j_i - 1)\alpha^{(6\ell+4)\kappa}$, where $j_i + 1$ is the number of leaves of p_i; therefore, for $C' = \bigcup C_i$ we have $\|\mathbf{z}_{C'}\|^2 \leq \sum \|\mathbf{z}_{C_i}\|^2 \leq (2n-3)\alpha^{(6\ell+4)\kappa}$ (we used that $\sum j_i \leq n-1$). This implies that $\|\mathbf{z}_{C \setminus C'}\|^2 \geq \|\mathbf{z}_C\|^2 - (2n-3)\alpha^{(6\ell+4)\kappa}$, and consequently $\mathbf{z}_C \preceq^{\ell}_w \mathbf{z}_{C \setminus C'}$, as otherwise all constraints containing z would have been removed in Step 2. Finally, observe that $B = (C \setminus C') +^{\ell} t_2$, and therefore $\mathbf{z}_{C \setminus C'} \preceq^{\ell} \mathbf{y}_B$ and, hence, $\mathbf{z}_{C \setminus C'} \preceq^{\ell}_w \mathbf{y}_B$. Putting this together with all other derived $\preceq^{\ell}_w$-relations, we obtain the required claim. ∎

Next, we move to proving the condition $(\mathrm{IPQ})_n$. For that, we will need the following technical statement. Intuitively, the statement says that, starting with a set A, if we follow a circular path of n-tree patterns and end up back in the set A, then all values from A can be reached by this pattern.

LEMMA 8.14. *Let $1 \leq \ell \leq |D|$, let p be a pattern from x to x which is a path of n-trees, and let A, B be such that $A +^{\ell} p = B$. If $B \cap D^{\ell}_x \subseteq A \cap D^{\ell}_x$, then $A \cap D^{\ell}_x = B \cap D^{\ell}_x$.*

Proof. For a contradiction, suppose that there is an element $a \in (D^{\ell}_x \cap A) \setminus B$. From Lemma 8.1, we get that $\|\mathbf{x}_{A \setminus B}\|^2 \geq \|\mathbf{x}_a\|^2 \geq 2\|\mathbf{x}_{D \setminus D^{\ell}_x}\|^2 \geq 2\|\mathbf{x}_{B \setminus A}\|^2$. Therefore, we have

$$\begin{aligned}\|\mathbf{x}_B\|^2 &= \|\mathbf{x}_A\|^2 - \|\mathbf{x}_{A \setminus B}\|^2 + \|\mathbf{x}_{B \setminus A}\|^2 \leq \|\mathbf{x}_A\|^2 - \|\mathbf{x}_a\|^2 + (1/2)\|\mathbf{x}_a\|^2 \\ &= \|\mathbf{x}_A\|^2 - (1/2)\|\mathbf{x}_a\|^2 \leq \|\mathbf{x}_A\|^2 - (1/2)\alpha^{6\ell\kappa}.\end{aligned}$$

On the other hand, since p is a path of n-trees, we get from the previous lemma that $\|\mathbf{x}_B\|^2 \geq \|\mathbf{x}_A\|^2 - \alpha^{(6\ell+2)\kappa}$. If we adjust constant c_0 from section 8.1 so that $1/2 > \alpha^{2\kappa}$, the above inequalities give a contradiction. ∎

The final lemma of this section proves a slight generalization of the condition $(\mathrm{IPQ})_n$.

LEMMA 8.15. *Let x be a variable, let p and q be two patterns from x to x which are paths of n-trees, let $1 \leq \ell \leq |D|$, and let $A \subseteq D^{\ell}_x$. Then there exists some j such that $A \subseteq A +^{\ell} (j(p+q)+p)$.*

Proof. For every A, define $A_0, A_1, \dots$ in the following way. If $i = 2j$ is even, then $A_i = A +^{\ell} (j(p+q))$. Otherwise, if $i = 2j+1$ is odd, then $A_i = A +^{\ell} (j(p+q)+p)$.

We claim that for every sufficiently large u we have $A_u \cap D^{\ell}_x = A_{u+1} \cap D^{\ell}_x$. From the finiteness of D, we get that for every sufficiently large u there is $u' > u$ such that $A_u = A_{u'}$. It follows that there exists some path of n-trees pattern p' starting and ending in x such that $A_u = A_{u+1} +^{\ell} p'$. To prove the claim, we will show that $\mathbf{x}_{A_u}$ and $\mathbf{x}_{A_{u+1}}$ are not ℓ-cut. Then the claim follows, as otherwise we would have removed all constraints involving x in Step 3.

Consider the path $x_1, \dots, x_k$ in p' which connects the beginning and end vertices. Further, let $R_i = R$ if the ith edge of the path is labeled by $((x_i, x_{i+1}), R)$, and let $R_i = R^{-1}$ if the ith edge is labeled by $((x_{i+1}, x_i), R)$. Now define a sequence $B_1, B'_2, B_2, \dots, B_m$ inductively by setting $B_1 = A_{u+1}$, $B'_{i+1} = B_i +^{\ell} (x_i, R_i, x_{i+1})$. Further, if x_{i+1} is not a branching vertex, then put $B_{i+1} = B'_{i+1}$. If x_{i+1} is a branching vertex, then let Φ_i be the set of all subtrees separated by x_{i+1} in p', excluding the two such subtrees containing the beginning and the end of p'. Then turn each subtree in Φ_i into a pattern by choosing x_{i+1} as the beginning and any other leaf as the end, and define $B_{i+1} = \{b \in B'_{i+1} : \{b\} +^{\ell} t \neq \emptyset \text{ for all } t \in \Phi_i\}$. As in Lemma 8.13, we know that the sum of the numbers of leaves of the trees from Φ_i that are also leaves of p' is less than $n-1$. Finally, if $\mathbf{x}_{A_u}$ and $\mathbf{x}_{A_{u+1}}$ are ℓ-cut, then, for some i, vectors $\mathbf{x}_{iB_i}$ and $\mathbf{x}_{i+1B'_{i+1}}$ are ℓ-cut, or vectors $\mathbf{x}_{iB_i}$ and $\mathbf{x}_{iB'_i}$ are ℓ-cut. The former case is impossible since $B'_{i+1} = B_i +^{\ell} (x_i, R_i, x_{i+1})$, and hence if $\mathbf{x}_{B'_{i+1}}$ and $\mathbf{x}_{B_i}$ are ℓ-cut, then either of the constraints $((x_i, x_{i+1}), R_i)$ or $((x_{i+1}, x_i), R^{-1})$ would have been removed in Step 4. We now show that the latter case is not possible either. Clearly, in this case x_i is a branching vertex. For $t \in \Phi_i$, let $C_t = \{b \in B'_i : \{b\} +^{\ell} t = \emptyset\}$ and let j_t be the number of leaves of t. By Lemma 8.12, we get $\|\mathbf{x}_{iC_t}\|^2 \leq (2j_t - 1)\alpha^{(6\ell+4)\kappa}$ for any $t \in \Phi_i$, and consequently

$$\|\mathbf{x}_{iB'_i} - \mathbf{x}_{iB_i}\|^2 \leq \sum_{t \in \Phi_i} \|\mathbf{x}_{iC_t}\|^2 \leq \sum_{t \in \Phi_i} (2j_t - 1)\alpha^{(6\ell+4)\kappa} \leq (2n-3)\alpha^{(6\ell+4)\kappa}.$$

Therefore, if $\mathbf{x}_{iB_i}$ and $\mathbf{x}_{iB'_i}$ were ℓ-cut, then all constraints that include x_i would have been removed in Step 5. We conclude that indeed we have $A_u \cap D^{\ell}_x = A_{u+1} \cap D^{\ell}_x$ for all sufficiently large u.

Now take $u = 2j+1$ large enough. We have that $(A \cup A_{u+1}) +^{\ell} (j(p+q)+p) = A_u \cup A_{2u+1}$. And also $(A_u \cup A_{2u+1}) \cap D^{\ell}_x = A_{u+1} \cap D^{\ell}_x \subseteq (A \cup A_{u+1}) \cap D^{\ell}_x$, and hence by Lemma 8.14 we get that $(A \cup A_{u+1}) \cap D^{\ell}_x = A_{u+1} \cap D^{\ell}_x$. Since $A \subseteq D^{\ell}_x$ by assumption of the lemma, we have $A \subseteq A_{u+1} \cap D^{\ell}_x \subseteq A_u = A +^{\ell} (j(p+q)+p)$. ∎

Finally, setting $A = \{a\}$ in Lemma 8.15 gives Proposition 5.3.

9. Full proof of Theorem 3.1(2). In this section, we prove Theorem 3.1(2). A brief outline of the proof is given in section 6. Throughout this section, $\mathcal{I} = (V, \mathcal{C})$ is a $(1-\varepsilon)$-satisfiable instance of CSP(Γ), where Γ consists of implicational constraints.

9.1. SDP relaxation. We use SDP relaxation (4.1)–(4.5) from section 4. For convenience, we write the SDP objective function as follows:

$$\sum_{C \in \mathcal{C} \text{ equals } (x=a) \vee (y=b)} w_C (\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b) \tag{9.1}$$

$$+ \frac{1}{2} \sum_{C \in \mathcal{C} \text{ equals } x=\pi(y)} \sum_{a \in D} w_C \|\mathbf{x}_{\pi(a)} - \mathbf{y}_a\|^2 + \sum_{C \in \mathcal{C} \text{ equals } x \in P} w_C \left(\sum_{a \in D \setminus P} \|\mathbf{x}_a\|^2 \right).$$

This expression equals (4.1) because of SDP constraint (4.4).

As discussed before (Lemma 4.1), we can assume that $\varepsilon \geq 1/m^2$, where m is the number of constraints. We solve the SDP with error $\delta = 1/m^2$ obtaining a solution, denoted by SDP, with objective value $O(\varepsilon)$. Note that every feasible SDP solution satisfies the following conditions:

$$\|\mathbf{x}_a\|^2 = \mathbf{x}_a \cdot \left(\mathbf{v}_0 - \sum_{b \neq a} \mathbf{x}_b\right) = \mathbf{x}_a \cdot \mathbf{v}_0 - \sum_{b \neq a} \mathbf{x}_a \cdot \mathbf{x}_b = \mathbf{x}_a \mathbf{v}_0, \tag{9.2}$$

$$\mathbf{x}_a \mathbf{y}_b = \mathbf{x}_a \cdot \left(\mathbf{v}_0 - \sum_{b' \neq b} \mathbf{y}_{b'}\right) = \|\mathbf{x}_a\|^2 - \sum_{b' \neq b} \mathbf{x}_a \mathbf{y}_{b'} \leq \|\mathbf{x}_a\|^2, \tag{9.3}$$

$$\|\mathbf{x}_a\|^2 - \|\mathbf{y}_b\|^2 = \|\mathbf{x}_a - \mathbf{y}_b\|^2 + 2(\mathbf{x}_a \mathbf{y}_b - \|\mathbf{y}_b\|^2) \leq \|\mathbf{x}_a - \mathbf{y}_b\|^2, \tag{9.4}$$

$$(\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b) = \sum_{a' \neq a} \mathbf{x}_{a'} \sum_{b' \neq b} \mathbf{y}_{b'} \geq 0. \tag{9.5}$$

9.2. Variable partitioning step. In this section, we describe the first step of our algorithm. In this step, we assign values to some variables, partition all variables into three groups $\mathcal{V}_0$, $\mathcal{V}_1$, and $\mathcal{V}_2$, and then split the instance into two subinstances $\mathcal{I}_1$ and $\mathcal{I}_2$.

Vertex partitioning procedure. Choose a number $r \in (0, 1/6)$ uniformly at random. Do the following for every variable x:

1. Let $D_x = \{a : 1/2 - r < \mathbf{x}_a \mathbf{v}_0\}$.
2. Depending on the size of D_x, do the following:
   (a) If $|D_x| = 1$, add x to $\mathcal{V}_0$ and assign $x = a$, where a is the single element of D_x.
   (b) If $|D_x| > 1$, add x to $\mathcal{V}_1$ and restrict x to D_x (see below for details).
   (c) If $D_x = \emptyset$, add x to $\mathcal{V}_2$.

Note that each variable in $\mathcal{V}_0$ is assigned a value; each variable x in $\mathcal{V}_1$ is restricted to a set D_x; each variable in $\mathcal{V}_2$ is not restricted.

LEMMA 9.1. (i) *If* $\mathbf{x}_a \mathbf{v}_0 > \frac{1}{2} + r$, *then* $x \in \mathcal{V}_0$.
(ii) *For every* $x \in \mathcal{V}_1$, $|D_x| = 2$.

Proof. (i) Note that for every $b \neq a$, we have $\mathbf{x}_a \mathbf{v}_0 + \mathbf{x}_b \mathbf{v}_0 \leq 1$ and, therefore, $\mathbf{x}_b \mathbf{v}_0 < 1/2 - r$. Hence, $b \notin D_x$. We conclude that $D_x = \{a\}$ and $x \in \mathcal{V}_0$.

(ii) Now consider $x \in \mathcal{V}_1$. We have

$$|D_x| < 3(1/2 - r)|D_x| = 3 \sum_{a \in D_x} (1/2 - r) \leq 3 \sum_{a \in D_x} \mathbf{x}_a \mathbf{v}_0 \leq 3.$$

Therefore, $|D_x| \leq 2$. Since $x \in \mathcal{V}_1$, $|D_x| > 1$. Hence, $|D_x| = 2$. ∎

We say that an assignment is admissible if it assigns a value in D_x to every $x \in \mathcal{V}_1$ and it is consistent with the partial assignment to variables in $\mathcal{V}_0$. From now on, we restrict our attention only to admissible assignments. We remove those constraints that are satisfied by every admissible assignment (our algorithm will satisfy all of them). Specifically, we remove the following constraints:

1. UG constraints $x = \pi(y)$ with $x, y \in \mathcal{V}_0$ that are satisfied by the partial assignment;
2. disjunction constraints $(x = a) \vee (y = b)$ such that either $x \in \mathcal{V}_0$ and x is assigned value a, or $y \in \mathcal{V}_0$ and y is assigned value b; and

3. unary constraints $x \in P$ such that either $x \in \mathcal{V}_0$ and the value assigned to x is in P, or $x \in \mathcal{V}_1$ and $D_x \subseteq P$.

We denote the set of satisfied constraints by $\mathcal{C}_s$. Let $\mathcal{C}' = \mathcal{C} \setminus \mathcal{C}_s$ be the set of remaining constraints. We now define a set of *violated* constraints—those constraints that we conservatively assume will not be satisfied by our algorithm (even though some of them might be satisfied by the algorithm). We say that a constraint $C \in \mathcal{C}'$ is violated if at least one of the following conditions holds:

1. C is a unary constraint on a variable $x \in \mathcal{V}_0 \cup \mathcal{V}_1$.
2. C is a disjunction constraint $(x = a) \vee (y = b)$, and either $x \notin \mathcal{V}_1$ or $y \notin \mathcal{V}_1$ (or both).
3. C is a disjunction constraint $(x = a) \vee (y = b)$, and $x, y \in \mathcal{V}_1$, and either $a \notin D_x$ or $b \notin D_y$ (or both).
4. C is a UG constraint $x = \pi(y)$, and at least one of the variables x, y is in $\mathcal{V}_0$.
5. C is a UG constraint $x = \pi(y)$, and one of the variables x, y is in $\mathcal{V}_1$ and the other is in $\mathcal{V}_2$.
6. C is a UG constraint $x = \pi(y)$, $x, y \in \mathcal{V}_1$ but $D_x \neq \pi(D_y)$.

We denote the set of violated constraints by $\mathcal{C}_v$ and let $\mathcal{C}'' = \mathcal{C}' \setminus \mathcal{C}_v$.

LEMMA 9.2. $\mathbb{E}[w(\mathcal{C}_v)] = O(\varepsilon)$.

Proof. We analyze separately constraints of each type in $\mathcal{C}_v$.

Unary constraints. A unary constraint $x \in P$ in $\mathcal{C}$ is violated if and only if $x \in \mathcal{V}_0 \cup \mathcal{V}_1$ and $D_x \not\subseteq P$ (if $D_x \subseteq P$, then $C \in \mathcal{C}_s$, and thus C is not violated). Thus, the SDP contribution of each violated constraint C of the form $x \in P$ is at least

$$w_C \sum_{a \in D \setminus P} \|\mathbf{x}_a\|^2 \geq w_C \sum_{a \in D_x \setminus P} \|\mathbf{x}_a\|^2 = w_C \sum_{a \in D_x \setminus P} \mathbf{x}_a \cdot \mathbf{v}_0 \geq w_C \left(\frac{1}{2} - r\right) \geq \frac{w_C}{3}.$$

The last two inequalities hold because the set $D_x \setminus P$ is nonempty, $\mathbf{x}_a \mathbf{v}_0 \geq 1/2 - r$ for all $a \in D_x$ by the construction, and $r \leq 1/6$. Therefore, the expected total weight of violated unary constraints is at most $3\,\mathsf{SDP} = O(\varepsilon)$.

Disjunction constraints. Consider a disjunction constraint $(x = a) \vee (y = b)$. Denote it by C. Assume without loss of generality that $\mathbf{x}_a \mathbf{v}_0 \geq \mathbf{y}_b \mathbf{v}_0$. Consider several cases. If $\mathbf{x}_a \mathbf{v}_0 > 1/2 + r$, then $x \in \mathcal{V}_0$ and x is assigned value a. Thus, C is satisfied. If $\mathbf{x}_a \mathbf{v}_0 \leq 1/2 + r$ and $\mathbf{y}_b \mathbf{v}_0 > 1/2 - r$, then we also have $\mathbf{x}_a \mathbf{v}_0 > 1/2 - r$, and hence $x, y \in \mathcal{V}_0 \cup \mathcal{V}_1$ and $a \in D_x$, $b \in D_y$. Thus, C is not violated (if at least one of the variables x and y is in $\mathcal{V}_0$, then $C \in \mathcal{C}_s$; otherwise, $C \in \mathcal{C}'$). Therefore, C is violated only if

$$\mathbf{x}_a \mathbf{v}_0 \leq 1/2 + r \quad \text{and} \quad \mathbf{y}_b \mathbf{v}_0 \leq 1/2 - r$$

or, equivalently,

$$\mathbf{x}_a \mathbf{v}_0 - 1/2 \leq r \leq 1/2 - \mathbf{y}_b \mathbf{v}_0. \tag{9.6}$$

Since we choose r uniformly at random in $(0, 1/6)$, the probability density of the random variable r is 6 on $(0, 1/6)$. Thus, the probability of event (9.6) is at most

$$6 \max\Big(\big((1/2 - \mathbf{y}_b \mathbf{v}_0) - (\mathbf{x}_a \mathbf{v}_0 - 1/2)\big), 0\Big) = 6 \max\Big((\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b) - \mathbf{x}_a \mathbf{y}_b, 0\Big)$$
$$\overset{\text{by (4.2) and (9.5)}}{\leq} 6(\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b).$$

The expected weight of violated constraints is at most

$$\sum_{\substack{C\in\mathcal{C} \text{ equals} \\ (x=a)\vee(y=b)}} 6w_C(\mathbf{v}_0-\mathbf{x}_a)(\mathbf{v}_0-\mathbf{y}_b) \le 6\,\mathsf{SDP} = O(\varepsilon).$$

UG constraints. Consider a UG constraint $x=\pi(y)$. Assume that it is violated. Then $D_x \neq \pi(D_y)$ (note that if x and y do not lie in the same set $\mathcal{V}_t$, then $|D_x| \neq |D_y|$ and necessarily $D_x \neq \pi(D_y)$). Thus, at least one of the sets $\pi(D_y)\setminus D_x$ or $D_x \setminus \pi(D_y)$ is not empty. If $\pi(D_y)\setminus D_x \neq \emptyset$, there exists $c \in \pi(D_y)\setminus D_x$. We have

$$\begin{aligned}\Pr\left(c\in\pi(D_y)\setminus D_x\right) &\le \Pr\left(\|\mathbf{y}_{\pi^{-1}(c)}\|^2 > 1/2-r \text{ and } \|\mathbf{x}_c\|^2 \le 1/2-r\right)\\ &= \Pr\left(1/2-\|\mathbf{y}_{\pi^{-1}(c)}\|^2 < r \le 1/2-\|\mathbf{x}_c\|^2\right)\\ &\le 6\max(\|\mathbf{y}_{\pi^{-1}(c)}\|^2-\|\mathbf{x}_c\|^2,0) \overset{\text{by (9.4)}}{\le} 6\|\mathbf{y}_{\pi^{-1}(c)}-\mathbf{x}_c\|^2.\end{aligned}$$

By the union bound, the probability that there is $c\in\pi(D_y)\setminus D_x$ is at most

$$6\sum_{c\in D}\|\mathbf{y}_{\pi^{-1}(c)}-\mathbf{x}_c\|^2 = 6\sum_{b\in D}\|\mathbf{y}_b-\mathbf{x}_{\pi(b)}\|^2.$$

Similarly, the probability that there is $b\in D_x\setminus\pi(D_y)$ is at most $6\sum_{b\in D}\|\mathbf{y}_b-\mathbf{x}_{\pi(b)}\|^2$. Therefore, the probability that the constraint $x=\pi(y)$ is violated is upper bounded by $12\sum_{b\in D}\|\mathbf{y}_b-\mathbf{x}_{\pi(b)}\|^2$. Consequently, the total expected weight of all violated UG constraints is at most

$$\begin{aligned}&\sum_{C\in\mathcal{C} \text{ equals } x=\pi(y)} w_C\left(12\sum_{b\in D}\|\mathbf{x}_{\pi(b)}-\mathbf{y}_b\|^2\right)\\ &\qquad = 24\times\left(\frac{1}{2}\sum_{C\in\mathcal{C} \text{ equals } x=\pi(y)} w_C\sum_{b\in D}\|\mathbf{x}_{\pi(b)}-\mathbf{y}_b\|^2\right)\\ &\qquad \le 24\,\mathsf{SDP} = O(\varepsilon),\end{aligned}$$

where we bound the value of the SDP by the second term of the objective function (9.1). ∎

We restrict our attention to the set $\mathcal{C}''$. There are four types of constraints in $\mathcal{C}''$:

1. disjunction constraints $(x=a)\vee(y=b)$ with $x,y\in\mathcal{V}_1$ and $a\in D_x$, $b\in D_y$;
2. UG constraints $x=\pi(y)$ with $x,y\in\mathcal{V}_1$ and $D_x=\pi(D_y)$;
3. UG constraints $x=\pi(y)$ with $x,y\in\mathcal{V}_2$; and
4. unary constraints $x\in P$ with $x\in\mathcal{V}_2$.

Denote the set of type 1 and 2 constraints by $\mathcal{C}_1$ and type 3 and 4 constraints by $\mathcal{C}_2$. Let $\mathcal{I}_1$ be the subinstance of $\mathcal{I}$ on variables $\mathcal{V}_1$ with constraints $\mathcal{C}_1$ in which every variable x is restricted to D_x, and let $\mathcal{I}_2$ be the subinstance of $\mathcal{I}$ on variables $\mathcal{V}_2$ with constraints $\mathcal{C}_2$.

In sections 9.3 and 9.4, we show how to solve $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. The total weight of constraints violated by our solution for $\mathcal{I}_1$ will be at most $O(\sqrt{\varepsilon})$. The total weight of constraints violated by our solution for $\mathcal{I}_2$ will be at most $O(\sqrt{\varepsilon\log|D|})$. Thus, the combined solution will satisfy a subset of the constraints of weight at least $1-O(\sqrt{\varepsilon\log|D|})$.

9.3. Solving instance $\mathcal{I}_1$. In this section, we present an algorithm that solves instance $\mathcal{I}_1$. The algorithm assigns values to variables in $\mathcal{V}_1$ so that the total weight of violated constraints is at most $O(\sqrt{\varepsilon})$.

LEMMA 9.3. *There is a randomized algorithm that, given instance $\mathcal{I}_1$ and the SDP solution $\{\mathbf{x}_a\}$ for $\mathcal{I}$, finds a set of UG constraints $\mathcal{C}_{bad} \subseteq \mathcal{C}_1$ and values $\alpha_x, \beta_x \in D_x$ for every $x \in \mathcal{V}_1$ such that the following conditions hold:*

- *$D_x = \{\alpha_x, \beta_x\}$.*
- *For each UG constraint $x = \pi(y)$ in $\mathcal{C}_1 \setminus \mathcal{C}_{bad}$, we have $\alpha_x = \pi(\alpha_y)$ and $\beta_x = \pi(\beta_y)$.*
- *The expected weight of $\mathcal{C}_{bad}$ is $O(\sqrt{\varepsilon})$.*

Proof. We use the algorithm of Goemans and Williamson for Min Uncut [26] to find values α_x, β_x. Recall that in the Min Uncut problem (also known as Min 2CNF$\equiv$ deletion) we are given a set of Boolean variables and a set of constraints of the form $(x = a) \leftrightarrow (y = b)$. Our goal is to find an assignment that minimizes the weight of unsatisfied constraints.

Consider the set of UG constraints in $\mathcal{C}_1$. Since $|D_x| = 2$ for every variable $x \in \mathcal{V}_1$, each constraint $x = \pi(y)$ is equivalent to the Min Uncut constraint $(x = \pi(a)) \leftrightarrow (y = a)$, where a is an element of D_y (it does not matter which of the two elements of D_y we choose). We define an SDP solution for the Goemans–Williamson relaxation of Min Uncut as follows. Consider $x \in \mathcal{V}_1$. Denote the elements of D_x by a and b (in any order). Let

$$\mathbf{x}_a^* = \frac{\mathbf{x}_a - \mathbf{x}_b}{\|\mathbf{x}_a - \mathbf{x}_b\|} \quad \text{and} \quad \mathbf{x}_b^* = -\mathbf{x}_a^* = \frac{\mathbf{x}_b - \mathbf{x}_a}{\|\mathbf{x}_a - \mathbf{x}_b\|}.$$

Note that the vectors $\mathbf{x}_a$ and $\mathbf{x}_b$ are nonzero orthogonal vectors, and thus $\|\mathbf{x}_a - \mathbf{x}_b\|$ is nonzero. The vectors $\mathbf{x}_a^*$ and $\mathbf{x}_b^*$ are unit vectors. Now we apply the random hyperplane rounding scheme of Goemans and Williamson: We choose a random hyperplane and let H be one of the half-spaces into which the hyperplane divides the space. Note that for every x exactly one of the two antipodal vectors in $\{\mathbf{x}_a^* : a \in D_x\}$ lies in H (almost surely). Define α_x and β_x so that $\mathbf{x}_{\alpha_x}^* \in H$ and $\mathbf{x}_{\beta_x}^* \notin H$. Let $\mathcal{C}_{\text{bad}}$ be the set of UG constraints such that $\alpha_x \neq \pi(\alpha_y)$, or equivalently $\mathbf{x}_{\pi(\alpha_y)}^* \notin H$.

Values α_x and β_x satisfy the first condition. If a UG constraint $x = \pi(y)$ is in $\mathcal{C}_1 \setminus \mathcal{C}_{\text{bad}}$, then $\alpha_x = \pi(\alpha_y)$; also since $D_x = \pi(D_y)$, $\beta_x = \pi(\beta_y)$. So the second condition holds. Finally, we verify the last condition. Consider a constraint $x = \pi(y)$. Let $\mathbf{A} = \mathbf{x}_{\pi(\alpha_y)} - \mathbf{x}_{\pi(\beta_y)}$ and $\mathbf{B} = \mathbf{y}_{\alpha_y} - \mathbf{y}_{\beta_y}$. Since $x \in \mathcal{V}_1$, we have $\|\mathbf{x}_{\pi(\alpha_y)}\|^2 > 1/2 - r > 1/3$ and $\|\mathbf{x}_{\pi(\beta_y)}\|^2 > 1/3$. Hence, $\|\mathbf{A}\|^2 = \|\mathbf{x}_{\pi(\alpha_y)}\|^2 + \|\mathbf{x}_{\pi(\beta_y)}\|^2 > 2/3$. Similarly, $\|\mathbf{B}\|^2 > 2/3$. Assume first that $\|\mathbf{A}\| \geq \|\mathbf{B}\|$. Then

$$\begin{aligned}\|\mathbf{x}_{\pi(\alpha_y)}^* - \mathbf{y}_{\alpha_y}^*\|^2 &= \left\| \frac{\mathbf{A}}{\|\mathbf{A}\|} - \frac{\mathbf{B}}{\|\mathbf{B}\|} \right\|^2 = 2 - \frac{2\mathbf{A}\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \\ &= \frac{2}{\|\mathbf{B}\|^2} \times \left(\|\mathbf{B}\|^2 - \frac{\|\mathbf{B}\|}{\|\mathbf{A}\|} \mathbf{A}\mathbf{B} \right).\end{aligned}$$

We have $2\big(\|\mathbf{B}\|^2 - \frac{\|\mathbf{B}\|}{\|\mathbf{A}\|} \mathbf{A}\mathbf{B}\big) \leq \|\mathbf{A} - \mathbf{B}\|^2$ since

$$\|\mathbf{A} - \mathbf{B}\|^2 - 2\left(\|\mathbf{B}\|^2 - \frac{\|\mathbf{B}\|}{\|\mathbf{A}\|} \mathbf{A}\mathbf{B} \right) = \Big(\|\mathbf{A}\| - \|\mathbf{B}\| \Big) \left(\|\mathbf{A}\| + \|\mathbf{B}\| - \frac{2\mathbf{A}\mathbf{B}}{\|\mathbf{A}\|} \right) \geq 0$$

because $\|\mathbf{A}\| \geq \mathbf{AB}/\|\mathbf{A}\|$ and $\|\mathbf{B}\| \geq \mathbf{AB}/\|\mathbf{A}\|$. We conclude that

$$\begin{aligned}\|\mathbf{x}^*_{\pi(\alpha_y)} - \mathbf{y}^*_{\alpha_y}\|^2 &\leq \frac{\|\mathbf{A}-\mathbf{B}\|^2}{\|\mathbf{B}\|^2} \leq \frac{3}{2}\|\mathbf{A}-\mathbf{B}\|^2 \\ &= \frac{3}{2}\left\|(\mathbf{x}_{\pi(\alpha_y)} - \mathbf{y}_{\alpha_y}) - (\mathbf{x}_{\pi(\beta_y)} - \mathbf{y}_{\beta_y})\right\|^2 \\ &\leq 3\left\|\mathbf{x}_{\pi(\alpha_y)} - \mathbf{y}_{\alpha_y}\right\|^2 + 3\left\|\mathbf{x}_{\pi(\beta_y)} - \mathbf{y}_{\beta_y}\right\|^2.\end{aligned}$$

If $\|\mathbf{A}\| \leq \|\mathbf{B}\|$, we get the same bound on $\|\mathbf{x}^*_{\pi(\alpha_y)} - \mathbf{y}^*_{\alpha_y}\|^2$ by swapping $\mathbf{A}$ and $\mathbf{B}$ in the formulas above. Therefore,

$$\sum_{\substack{C\in\mathcal{C}_1\\ \text{is of the form}\\ x=\pi(y)}} w_C\|\mathbf{x}^*_{\pi(\alpha_y)} - \mathbf{y}^*_{\alpha_y}\|^2 \leq 3\,\mathsf{SDP} = O(\varepsilon).$$

The analysis by Goemans and Williamson shows that the expected total weight of the constraints of the form $x = \pi(y)$ such that

$$\mathbf{x}^*_{\pi(\alpha_y)} \notin H \quad \text{and} \quad \mathbf{y}^*_{\alpha_y} \in H$$

is at most $O(\sqrt{\varepsilon})$; see section 3 in [26] for the original analysis or section 2 in survey [47] for a presentation more closely aligned with our notation. Therefore, the expected total weight of $\mathcal{C}_{\text{bad}}$ is $O(\sqrt{\varepsilon})$. ∎

We remove all constraints $\mathcal{C}_{\text{bad}}$ from $\mathcal{I}_1$ and obtain an instance $\mathcal{I}'_1$ (with the domain for each variable x now restricted to D_x). We construct an SDP solution $\{\tilde{\mathbf{x}}_a\}$ for $\mathcal{I}'_1$. We let

$$\tilde{\mathbf{x}}_{\alpha_x} = \mathbf{x}_{\alpha_x} \quad \text{and} \quad \tilde{\mathbf{x}}_{\beta_x} = \mathbf{v}_0 - \mathbf{x}_{\alpha_x}.$$

We define $S_{x\alpha_x} = \{\alpha_x\}$ and $S_{x\beta_x} = D \setminus S_{x\alpha_x}$. Since $\tilde{\mathbf{x}}_{\beta_x} = \mathbf{v}_0 - \mathbf{x}_{\alpha_x} = \mathbf{x}_{S_{x\beta_x}}$, we have

$$\tilde{\mathbf{x}}_a = \mathbf{x}_{S_{xa}} \quad \text{for every } a \in D_x. \tag{9.7}$$

Note that $a \in S_{xa}$ for every $a \in D_x$.

LEMMA 9.4. *The solution $\{\tilde{\mathbf{x}}_a\}$ is a feasible solution for SDP relaxation* (4.1)–(4.5) *for $\mathcal{I}'_1$. Its cost is $O(\varepsilon)$.*

Proof. We verify that the SDP solution is feasible. First, we have $\sum_{a\in D_x} \tilde{\mathbf{x}}_a = \mathbf{v}_0$ and

$$\tilde{\mathbf{x}}_{\alpha_x}\tilde{\mathbf{x}}_{\beta_x} = \mathbf{x}_{\alpha_x}\cdot(\mathbf{v}_0 - \mathbf{x}_{\alpha_x}) = \mathbf{x}_{\alpha_x}\mathbf{v}_0 - \|\mathbf{x}_{\alpha_x}\|^2 = 0.$$

Then, for $a \in D_x$ and $b \in D_y$, we have $\tilde{\mathbf{x}}_a\tilde{\mathbf{y}}_b = \sum_{a'\in S_{xa}, b'\in S_{yb}} \mathbf{x}_{a'}\mathbf{y}_{b'} \geq 0$. We now show that the SDP cost is $O(\varepsilon)$.

First, we consider disjunction constraints. We prove that the contribution of each constraint $(x = a) \vee (y = b)$ to the SDP for $\mathcal{I}'_1$ is at most its contribution to the SDP for $\mathcal{I}$. That is,

$$(\mathbf{v}_0 - \tilde{\mathbf{x}}_a)(\mathbf{v}_0 - \tilde{\mathbf{y}}_b) \leq (\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b). \tag{9.8}$$

Observe that $(\mathbf{v}_0 - \tilde{\mathbf{x}}_a) = \mathbf{x}_{D\setminus S_{xa}}$, $(\mathbf{v}_0 - \tilde{\mathbf{y}}_b) = \mathbf{y}_{D\setminus S_{yb}}$, $(\mathbf{v}_0 - \mathbf{x}_a) = \mathbf{x}_{D\setminus\{a\}}$, and $(\mathbf{v}_0 - \mathbf{y}_b) = \mathbf{y}_{D\setminus\{b\}}$. Then $D \setminus S_{xa} \subseteq D \setminus \{a\}$ and $D \setminus S_{yb} \subseteq D \setminus \{b\}$. Therefore, by (4.2),

$$\begin{aligned}(\mathbf{v}_0 - \tilde{\mathbf{x}}_a)(\mathbf{v}_0 - \tilde{\mathbf{y}}_b) &= \sum_{(a',b')\in(D\setminus S_{xa})\times(D\setminus S_{yb})} \mathbf{x}_{a'}\mathbf{y}_{b'} \leq \sum_{(a',b')\in(D\setminus\{a\})\times(D\setminus\{b\})} \mathbf{x}_{a'}\mathbf{y}_{b'} \\ &= (\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b).\end{aligned}$$

Now we consider UG constraints. The contribution of a UG constraint $x = \pi(y)$ in $\mathcal{C}_1 \setminus \mathcal{C}_{\text{bad}}$ to the SDP for $\mathcal{I}_1'$ equals the weight of the constraint times the following expression:

$$\begin{aligned}\|\tilde{\mathbf{x}}_{\pi(\alpha_y)} - \tilde{\mathbf{y}}_{\alpha_y}\|^2 + \|\tilde{\mathbf{x}}_{\pi(\beta_y)} - \tilde{\mathbf{y}}_{\beta_y}\|^2 &= \|\tilde{\mathbf{x}}_{\alpha_x} - \tilde{\mathbf{y}}_{\alpha_y}\|^2 + \|\tilde{\mathbf{x}}_{\beta_x} - \tilde{\mathbf{y}}_{\beta_y}\|^2 \\ &= \|\mathbf{x}_{\alpha_x} - \mathbf{y}_{\alpha_y}\|^2 + \|(\mathbf{v}_0 - \mathbf{x}_{\alpha_x}) - (\mathbf{v}_0 - \mathbf{y}_{\alpha_y})\|^2 \\ &= 2\|\mathbf{x}_{\alpha_x} - \mathbf{y}_{\alpha_y}\|^2 = 2\|\mathbf{x}_{\pi(\alpha_y)} - \mathbf{y}_{\alpha_y}\|^2.\end{aligned}$$

Thus, by the choice of α_x and α_y (Lemma 9.3), the contribution is at most twice the contribution of the constraint to the SDP for $\mathcal{I}$. We conclude that the SDP contribution of all the constraints in $\mathcal{C}_1 \setminus \mathcal{C}_{\text{bad}}$ is at most $2\,\mathsf{SDP} = O(\varepsilon)$. ∎

Finally, we note that $\mathcal{I}_1'$ is a Boolean 2-CSP instance. We round solution $\{\tilde{\mathbf{x}}_a\}$ using the rounding procedure by Charikar, Makarychev, and Makarychev for Boolean 2-CSP [17] (when $|D| = 2$, the SDP relaxation used in [17] is equivalent to SDP (4.1)–(4.5)). We get an assignment of variables in $\mathcal{V}_1$. The weight of constraints in $\mathcal{C}_1 \setminus \mathcal{C}_{\text{bad}}$ violated by this assignment is at most $O(\sqrt{\varepsilon})$. Since $w(\mathcal{C}_{\text{bad}}) = O(\sqrt{\varepsilon})$, the weight of constraints in $\mathcal{C}_1$ violated by the assignment is at most $O(\sqrt{\varepsilon})$.

9.4. Solving instance $\mathcal{I}_2$. Instance $\mathcal{I}_2$ is a UG instance with additional unary constraints. We restrict the SDP solution for $\mathcal{I}$ to variables $x \in \mathcal{V}_2$ and get a solution for the UG instance $\mathcal{I}_2$. Note that since we do not restrict the domain of variables $x \in \mathcal{V}_2$ to D_x, the SDP solution we obtain is feasible. The SDP cost of this solution is at most SDP. We round this SDP solution using a variant of the algorithm by Charikar, Makarychev, and Makarychev [16] that is presented in section 3 of the survey [47]; this variant of the algorithm does not need ℓ_2^2-triangle-inequality SDP constraints. Given a $(1-\varepsilon)$-satisfiable instance of Unique Games, the algorithm finds a solution with the weight of violated constraints at most $O(\sqrt{\varepsilon \log |D|})$. We remark that paper [16] considers only UG instances. However, in [16], we can restrict the domain of any variable x to a set S_x by setting $\mathbf{x}_a = 0$ for $a \in D \setminus S_x$. Hence, we can model unary constraints as follows. For every unary constraint $x \in P$, we introduce a dummy variable $z_{x,P}$ and restrict its domain to the set P. Then we replace each constraint $x \in P$ with the equivalent constraint $x = z_{x,P}$. The weight of the constraints violated by the obtained solution is at most $O(\sqrt{\varepsilon \log |D|})$.

Finally, we combine results proved in sections 9.2, 9.3, and 9.4 and obtain Theorem 3.1(2).

10. Conclusion. We have proved that every CSP with an NU polymorphism admits a robust algorithm with polynomial loss. Thus a small gap remains in our understanding of such algorithms—between the sufficient condition of having an NU polymorphism and a necessary condition SD(∨). We remark that closing this gap is likely to require a structural result, similar to our Theorem 5.1, which would resolve the conjecture of Larose and Tesson [44] and characterize CSPs solvable by linear propagation. Such a result would immediately imply a characterization of CSPs in the complexity class NL [21, 44] (and hence also L [33]), modulo complexity-theoretic assumptions.

REFERENCES

[1] P. Austrin and J. Håstad, *On the usefulness of predicates*, ACM Trans. Comput. Theory, 5 (2013), 1, https://doi.org/10.1145/2462896.2462897.

[2] K. Baker and A. Pixley, *Polynomial interpolation and the Chinese remainder theorem*, Math. Z., 143 (1975), pp. 165–174, https://doi.org/10.1007/BF01187059.
[3] L. Barto, *Finitely related algebras in congruence distributive varieties have near unanimity terms*, Canad. J. Math., 65 (2013), pp. 3–21, https://doi.org/10.4153/CJM-2011-087-3.
[4] L. Barto and M. Kozik, *Constraint satisfaction problems solvable by local consistency methods*, J. ACM, 61 (2014), 3, https://doi.org/10.1145/2556646.
[5] L. Barto and M. Kozik, *Robustly solvable constraint satisfaction problems*, SIAM J. Comput., 45 (2016), pp. 1646–1669, https://doi.org/10.1137/130915479.
[6] L. Barto, M. Kozik, and R. Willard, *Near unanimity constraints have bounded pathwidth duality*, in Proceedings of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), 2012, pp. 125–134, https://doi.org/10.1109/LICS.2012.24.
[7] L. Barto, A. Krokhin, and R. Willard, *Polymorphisms, and how to use them*, in The Constraint Satisfaction Problem: Complexity and Approximability, Dagstuhl Follow-Ups 7, A. Krokhin and S. Živný, eds., Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Wadern, Germany, 2017, pp. 1–44, https://doi.org/10.4230/DFU.Vol7.15301.1.
[8] W. Bibel, *Constraint satisfaction from a deductive viewpoint*, Artificial Intelligence, 35 (1988), pp. 401–413, https://doi.org/10.1016/0004-3702(88)90023-9.
[9] A. Bulatov, *Bounded Relational Width*, manuscript, 2009.
[10] A. Bulatov, *A dichotomy theorem for nonuniform CSPs*, in Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2017, pp. 319–330, https://doi.org/10.1109/FOCS.2017.37.
[11] A. Bulatov, P. Jeavons, and A. Krokhin, *Classifying complexity of constraints using finite algebras*, SIAM J. Comput., 34 (2005), pp. 720–742, https://doi.org/10.1137/S0097539700376676.
[12] A. Bulatov, A. Krokhin, and B. Larose, *Dualities for constraint satisfaction problems*, in Complexity of Constraints, Lecture Notes in Comput. Sci. 5250, Springer, Berlin, Heidelberg, 2008, pp. 93–124, https://doi.org/10.1007/978-3-540-92800-3_5.
[13] C. Carvalho, V. Dalmau, and A. Krokhin, *CSP duality and trees of bounded pathwidth*, Theoret. Comput. Sci., 411 (2010), pp. 3188–3208, https://doi.org/10.1016/j.tcs.2010.05.016.
[14] C. Carvalho, V. Dalmau, and A. Krokhin, *Two new homomorphism dualities and lattice operations*, J. Logic Comput., 21 (2011), pp. 1065–1092, https://doi.org/10.1093/logcom/exq030.
[15] S. O. Chan, *Approximation resistance from pairwise independent subgroups*, J. ACM, 63 (2016), 27, https://doi.org/10.1145/2873054.
[16] M. Charikar, K. Makarychev, and Y. Makarychev, *Near-optimal algorithms for unique games*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC), 2006, pp. 205–214, https://doi.org/10.1145/1132516.1132547.
[17] M. Charikar, K. Makarychev, and Y. Makarychev, *Near-optimal algorithms for maximum constraint satisfaction problems*, ACM Trans. Algorithms, 5 (2009), 32, https://doi.org/10.1145/1541885.1541893.
[18] D. Cohen and P. Jeavons, *The complexity of constraint languages*, in Handbook of Constraint Programming, F. Rossi, P. van Beek, and T. Walsh, eds., Elsevier, Amsterdam, 2006, pp. 245–280, https://doi.org/10.1016/S1574-6526(06)80012-X.
[19] M. Cooper, D. Cohen, and P. Jeavons, *Characterising tractable constraints*, Artificial Intelligence, 65 (1994), pp. 347–361, https://doi.org/10.1016/0004-3702(94)90021-3.
[20] N. Creignou, S. Khanna, and M. Sudan, *Complexity Classifications of Boolean Constraint Satisfaction Problems*, SIAM Monogr. Discrete Math. Appl. 7, SIAM, Philadelphia, 2001, https://doi.org/10.1137/1.9780898718546.
[21] V. Dalmau, *Linear Datalog and bounded path duality for relational structures*, Log. Methods Comput. Sci., 1 (2005), 1:5, https://doi.org/10.2168/LMCS-1(1:5)2005.
[22] V. Dalmau and A. Krokhin, *Majority constraints have bounded pathwidth duality*, European J. Combin., 29 (2008), pp. 821–837, https://doi.org/10.1016/j.ejc.2007.11.020.
[23] V. Dalmau and A. Krokhin, *Robust satisfiability for CSPs: Hardness and algorithmic results*, ACM Trans. Comput. Theory, 5 (2013), 15, https://doi.org/10.1145/2540090.
[24] V. Dalmau, A. Krokhin, and R. Manokaran, *Towards a characterization of constant-factor approximable Min CSPs*, in Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, Philadelphia, 2015, pp. 847–857, https://doi.org/10.1137/1.9781611973730.58.
[25] T. Feder and M. Y. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory*, SIAM J. Comput., 28 (1998), pp. 57–104, https://doi.org/10.1137/S0097539794266766.

[26] M. Goemans and D. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. Assoc. Comput. Mach., 42 (1995), pp. 1115–1145, https://doi.org/10.1145/227683.227684.
[27] V. Guruswami and Y. Zhou, *Tight bounds on the approximability of almost-satisfiable Horn SAT and exact hitting set*, Theory Comput., 8 (2012), pp. 239–267, https://doi.org/10.4086/toc.2012.v008a011.
[28] J. Håstad, *Some optimal inapproximability results*, J. ACM, 48 (2001), pp. 798–859, https://doi.org/10.1145/502090.502098.
[29] J. Håstad, *On the NP-hardness of Max-Not-*2, SIAM J. Comput., 43 (2014), pp. 179–193, https://doi.org/10.1137/120882718.
[30] D. Hobby and R. McKenzie, *The Structure of Finite Algebras*, Contemp. Math. 76, American Mathematical Society, Providence, RI, 1988.
[31] P. Jeavons, D. Cohen, and M. Cooper, *Constraints, consistency and closure*, Artificial Intelligence, 101 (1998), pp. 251–265, https://doi.org/10.1016/S0004-3702(98)00022-8.
[32] P. Jeavons, D. Cohen, and M. Gyssens, *Closure properties of constraints*, J. ACM, 44 (1997), pp. 527–548, https://doi.org/10.1145/263867.263489.
[33] A. Kazda, *n-permutability and linear Datalog implies symmetric Datalog*, Log. Methods Comput. Sci., 14 (2018), no. 2, 3, https://doi.org/10.23638/LMCS-14(2:3)2018.
[34] S. Khot, *On the power of unique* 2*-prover* 1*-round games*, in Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing (STOC), 2002, pp. 767–775, https://doi.org/10.1145/509907.510017.
[35] S. Khot, *On the unique games conjecture*, in Proceedings of the 25th Annual IEEE Conference on Computational Complexity (CCC), 2010, pp. 99–121, https://doi.org/10.1109/CCC.2010.19.
[36] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell, *Optimal inapproximability results for MAXCUT and other* 2*-variable CSPs?*, SIAM J. Comput., 37 (2007), pp. 319–357, https://doi.org/10.1137/S0097539705447372.
[37] S. Khot, M. Tulsiani, and P. Worah, *A characterization of strong approximation resistance*, in Proceedings of the Forty-Sixth ACM Symposium on Theory of Computing (STOC), 2014, pp. 634–643, https://doi.org/10.1145/2591796.2591817.
[38] V. Kolmogorov, A. Krokhin, and M. Rolínek, *The complexity of general-valued CSPs*, SIAM J. Comput., 46 (2017), pp. 1087–1110, https://doi.org/10.1137/16M1091836.
[39] V. Kolmogorov, J. Thapper, and S. Živný, *The power of linear programming for general-valued CSPs*, SIAM J. Comput., 44 (2015), pp. 1–36, https://doi.org/10.1137/130945648.
[40] M. Kozik, *Weaker Consistency Notions for all the CSPs of Bounded Width*, preprint, https://arxiv.org/abs/1605.00565v2, 2016.
[41] M. Kozik, A. Krokhin, M. Valeriote, and R. Willard, *Characterizations of several Maltsev conditions*, Algebra Universalis, 73 (2015), pp. 205–224, https://doi.org/10.1007/s00012-015-0327-2.
[42] A. Krokhin and S. Živný, eds., *The Constraint Satisfaction Problem: Complexity and Approximability*, Dagstuhl Follow-Ups 7, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Wadern, Germany, 2017, http://www.dagstuhl.de/dagpub/978-3-95977-003-3.
[43] G. Kun, R. O'Donnell, T. Suguru, Y. Yoshida, and Y. Zhou, *Linear programming, width-*1 *CSPs, and robust satisfaction*, in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS), 2012, pp. 484–495, https://doi.org/10.1145/2090236.2090274.
[44] B. Larose and P. Tesson, *Universal algebra and hardness results for constraint satisfaction problems*, Theoret. Comput. Sci., 410 (2009), pp. 1629–1647, https://doi.org/10.1016/j.tcs.2008.12.048.
[45] B. Larose, M. Valeriote, and L. Zádori, *Omitting types, bounded width and the ability to count*, Internat. J. Algebra Comput., 19 (2009), pp. 647–668, https://doi.org/10.1142/S021819670900524X.
[46] B. Larose and L. Zádori, *Bounded width problems and algebras*, Algebra Universalis, 56 (2007), pp. 439–466, https://doi.org/10.1007/s00012-007-2012-6.
[47] K. Makarychev and Y. Makarychev, *Approximation algorithms for CSPs*, in The Constraint Satisfaction Problem: Complexity and Approximability, Dagstuhl Follow-Ups 7, A. Krokhin and S. Živný, eds., Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017, pp. 287–325, https://doi.org/10.4230/DFU.Vol7.15301.11.
[48] P. Raghavendra, *Optimal algorithms and inapproximability results for every CSP?*, in Proceedings of the Fortieth ACM Symposium on Theory of Computing (STOC), 2008, pp. 245–254, https://doi.org/10.1145/1374376.1374414.
[49] T. Schaefer, *The complexity of satisfiability problems*, in Conference Record of the Tenth

Annual ACM Symposium on Theory of Computing (STOC), 1978, pp. 216–226, https://doi.org/10.1145/800133.804350.

[50] J. Thapper and S. Živný, *The complexity of finite-valued CSPs*, J. ACM, 63 (2016), 37, https://doi.org/10.1145/2974019.

[51] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Rev., 38 (1996), pp. 49–95, https://doi.org/10.1137/1038003.

[52] D. Zhuk, *A proof of CSP dichotomy conjecture*, in Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2017, pp. 331–342, https://doi.org/10.1109/FOCS.2017.38.

[53] U. Zwick, *Finding almost-satisfying assignments*, in Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC), 1998, pp. 551–560, https://doi.org/10.1145/276698.276869.