Robust algorithms with polynomial loss for near-unanimity CSPs*

Víctor Dalmau[†] Marcin Kozik[‡] Andrei Krokhin[§] Konstantin Makarychev[¶]
Yury Makarychev[‖] 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 CSP is called robust if it outputs an assignment satisfying a $(1 - g(\varepsilon))$-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 characterised 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. 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 dual discriminator with $k = 2$ for any domain size. In the latter case, the CSP is a common generalisation 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 CSP.

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 [17, 19, 26]. An instance of the CSP consists of a set of variables, a domain of values, and a set of constraints on combinations of 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. [10, 17, 19, 26]). 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 [8, 10, 26, 38, 50], pioneered by [49], which characterise 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 (e.g. [5, 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, e.g. [15, 16, 28, 35, 48]. 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) was much studied recently, e.g. [1, 14, 30, 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

*Marcin Kozik and Jakub Opršal were partially supported by the National Science Centre Poland under grant no. UMO-2014/13/B/ST6/01812; Jakub Opršal has also received funding from the European Research Council (Grant Agreement no. 681988, CSP-Infinity). Yury Makarychev was partially supported by NSF awards CAREER CCF-1150062 and IIS-1302662.

[†]University Pompeu Fabra
[‡]Jagiellonian University, Kraków
[§]Durham University
[¶]Microsoft Research
[‖]TTIC

exact quality of this approximation is unknown.

On the positive side, Zwick [52] 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 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 [52], Zwick gave an SDP-based robust algorithm with $g(\varepsilon) = O(\varepsilon^{1/3})$ for 2-Sat and 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 [16] for 2-Sat, and in [15] 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 [28]. These bounds for Horn k-Sat ($k \geq 3$), Horn 2-Sat, 2-Sat, and Unique Games(q) are known to be optimal [28, 34, 36], assuming the UGC.

The algebraic approach to CSP [10, 17, 33] 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 component-wise. 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. [4, 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 [10, 33], 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 [5, 41, 45] for details) equivalent to the inability to simulate Lin-p – systems of linear equations over Z_p, p prime, with 3 variable per equation. Håstad's celebrated result [29] implies that Lin-p does not admit a robust algorithm (for any g). This result carries over to all constraint languages that can simulate (some) Lin-p [23]. The remaining languages are precisely those that have the logico-combinatorial property of CSPs called "*bounded width*" or "*bounded treewidth duality*" [5, 9, 46]. This property says, roughly, that all unsatisfiable instances can be refuted via local propagation – see [11] for a survey on dualities for CSP. Barto and Kozik used SD($\wedge$) in [5], and then in [4] they used their techniques from [5] to prove the Guruswami-Zhou conjecture [28] that each bounded width CSP admits a robust algorithm.

The general bound on the loss in [4] 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, 42]. 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) [28]. The algebraic condition that separates 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].

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 (e.g. [26, 32, 6]). Bounded pathwidth duality for CSPs admitting an NU polymorphism was established in a series of papers [20, 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 *dual discriminator* (the corresponding CSP is a common generalisation of Unique Games with a fixed domain and 2-Sat), we can always choose $k = 2$. Our algorithms use the standard SDP relaxation for CSPs.

The algorithm for the general NU case is inspired by [4] and follows the same general scheme:

1. Solve the SDP relaxation for a $(1-\varepsilon)$-satisfiable instance $\mathcal{I}$.

2. Use the 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. 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 technical challenge in applying this scheme. Our algorithm is quite different from the algorithm in [4]. 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 $O(1/\log(1/\varepsilon))$ [28], unless UGC fails. To get 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 SPD relaxations that we will use further on. Sections 5 and 6 contain the description of the algorithms for constraint languages compatible with NU polymorphism and dual discriminator, respectively; the following chapters prove the correctness of the two algorithms. Proof of Theorem 5.1 is omitted due to space constraints, it can be found in the full version of this paper available on arXiv [21].

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 $\mathcal{C}$ is 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 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 $s : V \to D$. 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 Opt($\mathcal{I}$).

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

The *decision problem* for CSP(Γ) asks whether an input instance $\mathcal{I}$ of CSP(Γ) has an assignment satisfying all constraints in $\mathcal{I}$. The *optimization problem* for CSP(Γ) asks 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. Say, 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 CSP(Γ) depend only on the polymorphisms of Γ [10, 23, 33, 44].

An n-ary ($n \geq 3$) operation f is a *near-unanimity (NU)* operation if, for all $x, y \in D$, it satisfies

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

Note that the behaviour 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 [31]:

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

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

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

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

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].

NU polymorphisms appeared many times in the CSP literature. For example, they characterize the so-called "bounded strict width" property [26, 32], 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. *[26, 32] 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.*

A majority operation f is called the *dual discriminator* if $f(x, y, z) = x$ whenever x, y, z are pairwise distinct. Binary relations preserved the dual discriminator are known as *implicational* [7] or *0/1/all* [18] relations, they are the relations of one of four kinds:

1. relations $x = a \vee y = b$ for $a, b \in D$,
2. relations $x = \pi(y)$ where π is a permutation on D,
3. relations $P_1(x) \times P_2(y)$ where P_1 and P_2 are unary relations,
4. intersections 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 a near-unanimity polymorphism then* CSP(Γ) *admits a randomized 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 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 dependent 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 there is 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 properties), but it is unclear how far connections between the two notions go. 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. We remark that 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. This family includes languages not having an NU polymorphism of any arity – see [12, 13].

4 SDP relaxation

Associated to every instance $\mathcal{I} = (V, \mathcal{C})$ of 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. 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 weight given by the solution of the SDP to the pair (a,b) in C. The optimal SDP solution, then, gives as little weight 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 $\text{loss}(C) = \sum_{(a,b)\notin R} \mathbf{x}_a\mathbf{y}_b$. 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 $\text{SDPOpt}(\mathcal{I})$ be the optimum value of (4.1). It is clear that, for any instance $\mathcal{I}$, we have $\text{Opt}(\mathcal{I}) \geq \text{SDPOpt}(\mathcal{I}) \geq 0$. SDPs can be solved up to an arbitrarily small additive error ε' in time *poly*$(|\mathcal{I}|, \log(1/\varepsilon'))$ [51]. By letting ε' be exponentially small in $|\mathcal{I}|$, we may assume that we can find a solution to the SDP relaxation of value $(1 + 2^{-|\mathcal{I}|})\text{Opt}(\mathcal{I})$ (see Appendix A for details). Since we are only interested in the asymptotic performance of the algorithm, we can ignore the $2^{-|\mathcal{I}|}$ error term and assume that the value of the solution is at most $\text{Opt}(\mathcal{I})$.

5 Overview of the proof of Theorem 3.1(1)

We assume throughout that Γ has a near-unanimity polymorphism of arity $n+1$ ($n \geq 2$).

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 13 of [4], 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 then 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 is 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 as the disjoint union of p and q with 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 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 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 which is obtained as $t_1 + \cdots + t_j$ for some n-tree patterns $t_1, \ldots, t_j$.

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 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 non-empty sets D_x^ℓ where $1 \leq \ell \leq |D|+1$ and x runs through all variables in an instance $\mathcal{I}$. Let p be a pattern in $\mathcal{I}$ with beginning b and 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 $\mathrm{CSP}(\Gamma)$ over a set V of variables. We say that $\mathcal{I}$ satisfies condition $(\mathrm{IPQ})_n$ if the following holds:

$(\mathrm{IPQ})_n$ For every $y \in V$, there exist non-empty sets $D_y^1 \subseteq \ldots \subseteq D_y^{|D|} \subseteq D_y^{|D|+1} = D$ such that for any $x \in V$, any $\ell \leq |D|$, 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 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 $\mathrm{CSP}(\Gamma)$ satisfies $(IPQ)_n$, 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 $\mathrm{CSP}(\Gamma)$, removes $O(\varepsilon^{1/k})$ constraints from it to obtain a subinstance $\mathcal{I}'$ satisfying condition $(\mathrm{IPQ})_n$. 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 Preprocessing

The goals of preprocessing are: First, we deal with instances which are $(1-\varepsilon)$-satisfiable for $1/\varepsilon$ that is not bounded by a polynomial in the number of constraints. Second, we precompute the sets D_x^ℓ to be used for providing the $(\mathrm{IPQ})_n$ condition.

Let $\kappa = 1/k$ (we will often use κ to avoid overloading formulas). Assume that $\mathcal{C} = \{C_1, \ldots, C_m\}$ and that $w_{C_1} \geq w_{C_2} \geq \ldots \geq w_{C_m}$.

Preprocessing step 1. 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 5.1. *If $\varepsilon \leq 1/m^2$ then preprocessing step 1 returns an assignment that $(1-\sqrt{\varepsilon})$-satisfies $\mathcal{I}$.*

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

If preprocessing step 1 returns an assignment then we are done. So assume that it did not return an assignment. Then we know that $\varepsilon \geq 1/m^2$. We solve the SDP relaxation and obtain an optimal solution $\{\mathbf{x}_a\}$ $(x \in V, a \in D)$. We have that $\mathrm{SDPOpt}(\mathcal{I}) \leq \varepsilon$. Let $\alpha = \max\{\mathrm{SDPOpt}(\mathcal{I}), 1/m^2\}$. It is clear that $\alpha \leq \varepsilon$ and $\alpha^\kappa = O(\varepsilon^\kappa)$. Furthermore, this gives us that $1/\alpha \leq m^2$. This will be needed to argue that the main part of the algorithm runs in the polynomial time.

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

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

The sets D_x^ℓ are chosen such that for smaller ℓ's 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 elements in D_x^ℓ and outside.

5.3.2 Main part

Given 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 $s_\ell \in \{0, \ldots, m_0 - 1\}$

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 $\mathbf{x}_A \not\preceq_w^\ell \mathbf{x}_B$, then remove all the constraints in which x participates.

Step 3. For ever $1 \leq \ell \leq |D|$ 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 there 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 there exist subsets A, B such that $\|\mathbf{x}_A - \mathbf{x}_B\| \leq (2n-3)^{1/2}\alpha^{(3\ell+2)\kappa}$ and $\mathbf{x}_A$ and $\mathbf{x}_B$ are ℓ-cut, remove all constraints in which x participates.

Step 6. By Proposition 5.2 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 such that, according to the SDP solution, these constraints have a high probability to be violated. Intuitively, steps 1 and 2 ensure that a loss in $\|\mathbf{x}_A\|$ after propagation 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 7.13), and then that by following a path of n-trees we do not lose too much (step 2). This is needed in order for $\{a\} +^\ell (j(p+q)+p)$ to be non-vanishing as j increases. Steps 3–5 ensure that if A and B are connected by paths of n-trees in both directions (i.e. $A = B + p_1$ and $B = A + p_2$), hence $\mathbf{x}_A$ and $\mathbf{x}_B$ do not differ too much, then $A \cap D_x^\ell = B \cap D_x^\ell$. 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) we don't cross a borderline 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. Providing the rest of condition $(\text{IPQ})_n$.

The algorithm runs in polynomial time. Since D is fixed, it is clear that the 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 the two following propositions whose proof can be found in Section 7. These propositions show that our new consistency notion satisfies the requirements of Step 2 of the general scheme (for designing robust approximation algorithms).

PROPOSITION 5.1. *The expected total weight of constraints removed by the algorithm is $O(\varepsilon^\kappa)$.*

PROPOSITION 5.2. *The instance $\mathcal{I}'$ obtained after steps 0–5 satisfies the condition $(IPQ)_n$ (with the sets D_x^ℓ computed in preprocessing step 2).*

6 Overview of the proof of Theorem 3.1(2)

Note that a dual discriminator is a majority, hence every relation in Γ is 2-decomposable. Therefore, it follows, e.g. from Lemma 1 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 non-empty 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 does not depend on ε and $|D|$).

We now give an informal and somewhat imprecise sketch of the algorithm and its analysis. We present details in Section 8. 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.

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 3 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 4 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 8.2).

The constraints from the 4 groups above naturally form two disjoint sub-instances 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 et al. [16]. This algorithm finds a solution violating an $O(\sqrt{\varepsilon})$ fraction of all constraints if the optimal solution violates at most ε fraction of all constraints or $\text{SDPOpt}(\mathcal{I}_1) \leq \varepsilon$. 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 [27]. 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 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 8.4). The same is true for all but $O(\sqrt{\varepsilon})$ 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 [16] 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 $\mathcal{I}_2$ at the preprocessing step. We run the approximation algorithm for Unique Games by Charikar et al. [15] 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 [15], the weight of constraints violated by the algorithm [15] 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 preprocessing step to the variables from the set $\mathcal{V}_0$.

7 Full proof of Theorem 3.1(1)

In this subsection we prove Propositions 5.1 and 5.2. The following equalities, which can be directly verified, are used repeatedly in this section: for any subsets A, B of D, it holds that $\|\mathbf{x}_A\|^2 = \mathbf{x}_A \mathbf{y}_D$ and $\|\mathbf{y}_B - \mathbf{x}_A\|^2 =$

$\mathbf{x}_{D\setminus A}\mathbf{y}_B + \mathbf{x}_A\mathbf{y}_{D\setminus B}$.

7.1 Analysis of preprocessing steps

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 7.1. *There exists a constant $c > 0$ such that the sets $D_x^\ell \subseteq D$, $x \in V$, $1 \le \ell \le |D|$, obtained in Preprocessing step 2, are non-empty 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 non-empty. 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 $|D|$.

7.2 Proof of Proposition 5.1

LEMMA 7.2. *The total weight of the constraints removed at step 0 is at most α^κ.*

Proof. Follows from Lemma 3.3 of [52].

LEMMA 7.3. *Let $((x, y), R)$ be a constraint not removed at 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 $c > 0$. 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 at 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 7.1(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 $(6\ell + 6)\kappa \le 1 - \kappa$, and hence we conclude that $\mathbf{x}_A\mathbf{y}_{D\setminus B} \le (d + 1)\alpha^{(6\ell+6)\kappa}$. Then, we have that

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

The proof of the second case is identical.

LEMMA 7.4. *The expected weight of the constraints removed at step 1 is $O(\alpha^\kappa)$.*

Proof. Let $((x, y), R)$ be a constraint not removed at step 0. We shall see that the probability that it is removed at 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 7.3 that $\|\mathbf{y}_B\|^2 \ge \|\mathbf{x}_A\|^2 - d\alpha^{(6\ell+6)\kappa}$ for some constant $d > 0$. Hence, the probability that a value r_ℓ in step 1 makes that $\mathbf{y}_B \not\succeq^\ell \mathbf{x}_A$ is at most

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

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 7.5. *There exist constants $c, d > 0$ such that for every pair of vectors $\mathbf{x}_A$ and $\mathbf{y}_B$, the probability, p, that a unit vector $\mathbf{u}$ chosen uniformly at random cuts $\mathbf{x}_A$ and $\mathbf{y}_B$ satisfies*

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

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

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

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 7.6. *If the probability that all constraints involving a variable x are removed in Step 2, Step 3, or Step 5 is at most $c\alpha^\kappa$ for some constant $c > 0$, then the total expected weight of constraints removed this way 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

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

and the lemma is proved.

LEMMA 7.7. *The expected weight of the constraints removed at step 2 is $O(\alpha^\kappa)$.*

Proof. Let x be a variable. We shall prove that the probability that we remove all constraints involving x at step 2 is at most $c\alpha^\kappa$ for some constant $c > 0$, the rest is Lemma 7.6. Suppose that $A \subseteq B$ are such that $\|\mathbf{x}_B\|^2 - \|\mathbf{x}_A\|^2 = \|\mathbf{x}_B - \mathbf{x}_A\|^2 \le (2n-3)\alpha^{(6\ell+4)\kappa}$. 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

$$(2n-3)/m_0 \le (2n-3)/(\alpha^{-2\kappa} - 1) \le c_1\alpha^\kappa$$

for $\alpha^\kappa < 1/2$. Therefore, the probability that this happens for at least one pair of sets A, B is at most $2^{2|D|}c_1\alpha^\kappa = c\alpha^\kappa$.

LEMMA 7.8. *The expected weight of the constraints removed at step 3 is $O(\alpha^\kappa)$.*

Proof. According to Lemma 7.6, it is enough to prove that the probability that we remove all constraints involving x at step 3 is at most $c\alpha^\kappa$ for some constant c. Let A and B such that $A \cap D_x^\ell \ne B \cap D_x^\ell$. Let a be an element in symmetric difference $(A \cap D_x^\ell) \triangle (B \cap D_x^\ell)$. Then we have $\|\mathbf{x}_B - \mathbf{x}_A\| = \sqrt{\mathbf{x}_{D\setminus A}\mathbf{x}_B + \mathbf{x}_A \mathbf{x}_{D\setminus B}} \ge \|\mathbf{x}_a\| \ge \alpha^{3\ell\kappa}$, where the last inequality is by Lemma 7.1(1). Then by Lemma 7.5 the probability that $\mathbf{x}_A$ and $\mathbf{x}_B$ are not ℓ-cut is at most

$$(1-\alpha^{3\kappa\ell})^{m_\ell} \le \frac{1}{\exp(\alpha^{3\kappa\ell}m_\ell)} \le \frac{1}{\exp(\alpha^{-\kappa})} \le \alpha^\kappa.$$

Taking the union bound for all sets A, B and all values of ℓ we obtain the desired bound.

LEMMA 7.9. *The expected weight of the constraints removed at step 4 is $O(\alpha^\kappa)$.*

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

Fix ℓ and A, B such that $B = A +^\ell (x, R, y)$ and $\mathbf{y}_B \preceq^\ell \mathbf{x}_A$. Since $B = A +^\ell p$ we have that $\mathbf{x}_A \mathbf{y}_{D\setminus B} \le c_1\alpha^{(6\ell+6)\kappa}$, as shown in the proof of Lemma 7.3. Since $\|\mathbf{x}_A\|^2 = \mathbf{x}_A(\mathbf{y}_B + \mathbf{y}_{D\setminus B})$, it follows that $\mathbf{x}_A\mathbf{y}_B \ge \|\mathbf{x}_A\|^2 - c_1\alpha^{(6\ell+6)\kappa}$.

Also, we have $\|\mathbf{y}_B\|^2 = (\mathbf{x}_A\mathbf{y}_B + \mathbf{x}_{D\setminus A}\mathbf{y}_B)$ 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 $\alpha^{(6\ell+4)\kappa} + c_1\alpha^{(6\ell+6)\kappa} \le (c_1+1)\alpha^{(6\ell+4)\kappa}$.

Putting the bounds together, we have that

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

for some constant $c_2 > 0$.

Applying the union bound and Lemma 7.5 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 7.10. *The expected weight of the constraints removed at step 5 is $O(\alpha^\kappa)$.*

Proof. Again, according to Lemma 7.6, it is enough to prove that the probability that we remove all constraints involving x at step 5 is at most $c\alpha^\kappa$ for some constant c. Suppose that A, B are such that $\|\mathbf{x}_A - \mathbf{x}_B\| \le (2n-3)^{1/2}\alpha^{(3\ell+2)\kappa}$. Hence, by Lemma 7.5, the probability that $\mathbf{x}_A$ and $\mathbf{x}_B$ are ℓ-cut is at most

$$\begin{aligned}
&1 - (1 - c(2n-3)\alpha^{(3\ell+2)\kappa})^{m_\ell} \le \\
&1 - (1 - m_\ell c(2n-3)\alpha^{(3\ell+2)\kappa}) = m_\ell c(2n-3)\alpha^{(3\ell+2)\kappa} \le \\
&c(2n-3)\alpha^\kappa + c(2n-3)\alpha^{(3\ell+2)\kappa} \le c'(2n-3)\alpha^\kappa.
\end{aligned}$$

Taking the union bound for all sets A, B and all values of ℓ, we obtain the desired bound.

7.3 Proof of Proposition 5.2

All patterns appearing in this subsection are in $\mathcal{I}'$.

LEMMA 7.11. *Let $1 \le \ell \le |D|$, 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, $\|\mathbf{x}_A\| \le \|\mathbf{y}_B\| + \alpha^{(6\ell+4)\kappa}$.*

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.

Lemma 7.12. *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 7.11. Suppose then that p is a tree pattern with $j+1 > 2$ leaves and the statement is true for any $i < j$. Choose y to be the first branching vertex when following p from x, and let p_0 be the subpath of p starting at x and ending at y, further let $t_1, \ldots, t_h$ be all the (maximal) subtrees of p starting at y excluding p_0 (for each of them choose any other leaf as the end vertex). 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 $B = \bigcup_{i=1}^n B_i$, we have

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

Finally, since $A +^\ell p = \emptyset$ then $A +^\ell p_0 \subseteq B$, and the claim follows from Lemma 7.11.

Lemma 7.13. *Let $1 \leq \ell \leq |D|$, let p be a pattern from x to y which is a path of n-trees. If $A, B \subseteq D$ such that $B +^\ell p = A$, then $\|\mathbf{y}_A\|^2 \geq \|\mathbf{x}_B\|^2 - \alpha^{(6\ell+2)\kappa}$.*

Proof. We will prove that for any n-tree pattern t and A, B with $B +^\ell t = A$, we have $\mathbf{x}_B \preceq_w^\ell \mathbf{y}_A$, 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, and observe that t is not a path (see Lemma 7.11). Let v_x and v_y denote the beginning and the end vertex 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. Now, the vertex v_z separates t into several subtrees, namely t_1, a tree connecting v_x and v_z, t_2, a path connecting v_z and v_y, and several trees $p_1, \ldots, p_j$ which have v_z as one vertex and are disjoint with the path connecting v_x and v_y. For p_i we choose v_z to be the beginning, and any other leaf to be the end. Further, we know that for $C = B +^\ell t_1$ we have $\mathbf{x}_B \preceq_w^\ell \mathbf{z}_C$. Now, let $C_i = \{a \in D_z^{\ell+1} : a +^\ell p_i = \emptyset\}$. Then by Lemma 7.12, 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 $\|\mathbf{z}_{C'}\|^2 \leq \sum \|\mathbf{z}_{C_i}\|^2 \leq (2n-3)\alpha^{(6\ell+4)\kappa}$. This implies that $\|\mathbf{z}_{C\setminus C'}\|^2 \geq \|\mathbf{z}_C\|^2 - (2n-3)\alpha^{(6\ell+4)\kappa}$, and consequently $\mathbf{z}_C \preceq_w^\ell \mathbf{z}_{C\setminus C'}$ as otherwise all constraints containing z would have been removed at step 2. Finally, observe that $A = (C \setminus C') +^\ell t_2$, and therefore $\mathbf{z}_{C\setminus C'} \preceq^\ell \mathbf{y}_A$. Putting this together with all other derived $\preceq_w^\ell$-relations, we get the required claim.

Lemma 7.14. *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 $B +^\ell p = A$. If $A \cap D_x^\ell \subseteq B \cap D_x^\ell$ then $A \cap D_x^\ell = B \cap D_x^\ell$.*

Proof. For a contradiction, suppose that there is an element $a \in (D_x^\ell \cap B) \setminus A$. From Lemma 7.1, conditions (3) and (1) we get that $\|\mathbf{x}_{B\setminus A}\|^2 \geq \|\mathbf{x}_a\|^2 \geq 2\|\mathbf{x}_{D\setminus D_x^\ell}\|^2 \geq 2\|\mathbf{x}_{A\setminus B}\|^2$. Therefore, we have

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

On the other hand, since p is a path of n-trees, we get from the previous lemma that $\|\mathbf{x}_A\|^2 \geq \|\mathbf{x}_B\|^2 - \alpha^{(6\ell+2)\kappa}$. If we adjust constant c_0 from Section 7.1 so that $1/2 > \alpha^{2\kappa}$, the above inequalities give a contradiction.

Lemma 7.15. *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 $A \subseteq D_x^\ell$. Then there exists some j such that $A \subseteq A +^\ell (j(p+q)+p)$.*

Proof. For every A, define $A_0, A_1, \ldots$ 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 $A_u \cap D_x^\ell = A_{u+1} \cap D_x^\ell$. 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 at step 3.

Consider the path in p' which connects the beginning and end vertices of p', and let $v_1, \ldots, v_u$ be the vertices which appear on the path in this order with v_1 being the beginning and v_u being the end vertex of p'. Further, let $R_i = R$ if the i-th edge of the path is concurrent and labeled by $((x_i, x_{i+1}), R)$, and let $R_i = R^{-1}$ if the i-th edge is not concurrent and labeled by $((x_{i+1}, x_i), R)$. Now define a sequence $B_1, B_2', B_2, \ldots, B_m$ 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, 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 of p' starting at x_i except those two that contain (parts of) p'', and define $B_{i+1} = \{b \in B'_{i+1} : b +^\ell t \neq \emptyset$ for all $t \in \Phi_i\}$. Since p' is a path of n-trees, we know that the sum of the numbers of leaves of the trees from Φ_i that are also leaves of p' is strictly less then $n-1$. Finally, if $\mathbf{x}_{A_u}$ are $\mathbf{x}_{A_{u+1}}$ 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 first 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 at step 4. The second case is impossible, since from Lemma 7.12 we get $\|\mathbf{x}_{iC_t}\|^2 \leq (2j_t - 1)\alpha^{(6\ell+4)\kappa}$ for any $t \in \Phi_i$, $C_t = \{b \in B'_i : b +^\ell t = \emptyset\}$, and j_t being the number of leaves of t, 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 at step 5. We conclude that indeed we have $A_u \cap D_x^\ell = A_{u+1} \cap D_x^\ell$ 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 $(A_u \cup A_{2u+1}) \cap D_x^\ell = A_{u+1} \cap D_x^\ell \subseteq (A \cup A_{u+1}) \cap D_x^\ell$, hence by Lemma 7.14 we get that $(A \cup A_{u+1}) \cap D_x^\ell = A_{u+1} \cap D_x^\ell$. Since $A \subseteq D_x^\ell$ by assumption of the lemma, we have $A \subseteq A_{u+1} \cap D_x^\ell \subseteq A_u = A +^\ell (j(p+q)+p)$.

8 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.

8.1 SDP Relaxation We use (essentially) the same SDP relaxation of the problem as in Section 4. Minimize

$$\sum_{C\in\mathcal{C} \text{ equals } (x=a)\vee(y=b)} w_C(\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b) + \frac{1}{2}\sum_{C\in\mathcal{C} \text{ equals } x=\pi(y)} \sum_{a\in D} w_C\|\mathbf{x}_{\pi(a)} - \mathbf{y}_a\|^2 + \sum_{C\in\mathcal{C} \text{ equals } x\in P} w_C\left(\sum_{a\in D\setminus P}\|\mathbf{x}_a\|^2\right) \tag{8.1}$$

subject to

$$\mathbf{x}_a\mathbf{y}_b \geq 0 \qquad x, y \in V,\ a, b \in D \tag{8.2}$$

$$\mathbf{x}_a\mathbf{x}_b = 0 \qquad x \in V,\ a, b \in D,\ a \neq b \tag{8.3}$$

$$\sum_{a\in D} \mathbf{x}_a = \mathbf{v}_0 \qquad x \in V \tag{8.4}$$

$$\|\mathbf{x}_a - \mathbf{z}_c\|^2 \leq \|\mathbf{x}_a - \mathbf{y}_b\|^2 + \|\mathbf{y}_b - \mathbf{z}_c\|^2 \qquad x, y, z \in V,\ a, b, c \in D \tag{8.5}$$

$$\|\mathbf{v}_0\|^2 = 1. \tag{8.6}$$

We solve the relaxation and find an optimal SDP solution $\{\mathbf{x}_a\}$. Denote SDPOpt($\mathcal{I}$) by SDP. We have, $\mathsf{SDP} \leq \varepsilon$. Note that every feasible SDP solution satisfies the following conditions.

$$\|\mathbf{x}_a\|^2 = \mathbf{x}_a \cdot \big(\mathbf{v}_0 - \sum_{b\neq a} \mathbf{x}_b\big) = \mathbf{x}_a\mathbf{v}_0, \tag{8.7}$$

$$\begin{aligned} \mathbf{x}_a\mathbf{y}_b &= \mathbf{x}_a \cdot (\mathbf{v}_0 - \sum_{b'\in D\setminus\{b\}} \mathbf{y}_{b'}) \\ &= \|\mathbf{x}_a\|^2 - \sum_{b'\in D\setminus\{b\}} \mathbf{x}_a\mathbf{y}_{b'} \\ &\leq \|\mathbf{x}_a\|^2, \end{aligned} \tag{8.8}$$

$$\begin{aligned} \|\mathbf{x}_a\|^2 - \|\mathbf{y}_b\|^2 &= \|\mathbf{x}_a - \mathbf{y}_b\|^2 + 2(\mathbf{x}_a\mathbf{y}_b - \|\mathbf{y}_b\|^2) \\ &\leq \|\mathbf{x}_a - \mathbf{y}_b\|^2, \end{aligned} \tag{8.9}$$

$$(\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b) = \sum_{a'\neq a} \mathbf{x}_{a'} \sum_{b'\neq b} \mathbf{y}_{b'} \geq 0. \tag{8.10}$$

8.2 Preprocessing 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 sub-instances $\mathcal{I}_1$ and $\mathcal{I}_2$.

Preprocessing Step 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 $D_x = \varnothing$, 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 8.1. *(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 $x \in \mathcal{V}_1$, $|D_x| > 1$. We get, $|D_x| = 2$.

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;

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

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

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

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 $3\,\mathsf{SDP} \leq 3\varepsilon$.

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

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

or equivalently,

$$\mathbf{x}_a\mathbf{v}_0 - 1/2 \leq r \leq 1/2 - \mathbf{y}_b\mathbf{v}_0. \tag{8.11}$$

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 (8.11) is at most

$$\begin{aligned}&6\max\Big(\big((1/2-\mathbf{y}_b\mathbf{v}_0) - (\mathbf{x}_a\mathbf{v}_0 - 1/2)\big), 0\Big)\\ &\quad= 6\max\Big((\mathbf{v}_0-\mathbf{x}_a)(\mathbf{v}_0-\mathbf{y}_b) - \mathbf{x}_a\mathbf{y}_b, 0\Big)\\ &\quad\overset{\text{by (8.1) and (8.10)}}{\leq} 6(\mathbf{v}_0-\mathbf{x}_a)(\mathbf{v}_0-\mathbf{y}_b).\end{aligned}$$

The expected weight of violated constraints is at most,

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

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 one 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 $\pi(D_y) \setminus D_x \neq \varnothing$, there exists $b \in \pi(D_y) \setminus D_x$. We have,

$$\begin{aligned} P_b &\equiv \Pr\left(b \in \pi(D_y) \setminus D_x\right) \\ &\leq \Pr\left(\|\mathbf{y}_b\|^2 > 1/2 - r \text{ and } \|\mathbf{x}_{\pi(b)}\|^2 \leq 1/2 - r\right) \\ &= \Pr\left(1/2 - \|\mathbf{y}_b\|^2 < r \leq 1/2 - \|\mathbf{x}_{\pi(b)}\|^2\right) \\ &\leq 6 \max(\|\mathbf{y}_b\|^2 - \|\mathbf{x}_{\pi(b)}\|^2, 0) \\ &\leq 6\|\mathbf{y}_b - \mathbf{x}_{\pi(b)}\|^2. \end{aligned}$$

By the union bound, the probability that there is $b \in \pi(D_y) \setminus D_x$ is at most $6\sum_{b\in D} \|\mathbf{y}_b - \mathbf{x}_{\pi(b)}\|^2$. 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 weight of the violated UG constraints is at most $24\,\mathsf{SDP} = O(\varepsilon)$, in expectation.

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 $x, y \in \mathcal{V}_2$;
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 sub-instance of $\mathcal{I}$ on variables $\mathcal{V}_1$ with constraints $\mathcal{C}_1$ in which every variable x is restricted to D_x, and $\mathcal{I}_2$ be the sub-instance of $\mathcal{I}$ on variables $\mathcal{V}_2$ with constraints $\mathcal{C}_2$.

In Sections 8.3 and 8.4, 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 $1 - O(\sqrt{\varepsilon \log |D|})$.

8.3 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 8.3. *There is a randomized algorithm that given instance $\mathcal{I}_1$ and the SDP solution 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 constraints $x = \pi(y)$ in $\mathcal{C}_1 \setminus \mathcal{C}_{bad}$, we have $\alpha_x = \pi(\alpha_y)$ and $\beta_x = \pi(\beta_y)$.*
- $\mathbb{E}[w(\mathcal{C}_{bad})] \leq O(\sqrt{\varepsilon})$.

Proof. We use the algorithm of Goemans and Williamson for Min Uncut [27] to find values α_x, β_x. Recall that in the Min Uncut problem (also known as Min 2CNF≡ 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 the hyperplane divides the space into. 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 $\mathcal{C}_{bad}$ 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 $\mathcal{C}_1 \setminus \mathcal{C}_{bad}$, 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 $\|A\| \geq \|B|$. Then,

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

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

$$\begin{aligned}&\|\mathbf{A} - \mathbf{B}\|^2 - 2\Big(\|\mathbf{B}\|^2 - \frac{\|\mathbf{B}\|}{\|\mathbf{A}\|}\mathbf{AB}\Big)\\ &\qquad = \Big(\|\mathbf{A}\| - \|\mathbf{B}\|\Big)\Big(\|\mathbf{A}\| + \|\mathbf{B}\| - \frac{2\mathbf{AB}}{\|\mathbf{A}\|}\Big) \geq 0,\end{aligned}$$

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

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

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

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

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

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

is at most $O(\sqrt{\varepsilon})$, see [27] (or Lemma 7.5 in this paper for a similar argument). Therefore, $\mathbb{E}[w(\mathcal{C}_{bad})] \leq O(\sqrt{\varepsilon})$.

We remove all constraints $\mathcal{C}_{bad}$ from $\mathcal{I}_1$ and obtain an instance $\mathcal{I}'_1$. Now 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 $\tilde{\mathbf{x}}_{\beta_x} = \mathbf{v}_0 - \mathbf{x}_{\alpha_x} = \sum_{a \in S_{x\beta_x}} \mathbf{x}_a$, we have,

$$\tilde{\mathbf{x}}_a = \sum_{a' \in S_{xa}} \mathbf{x}_{a'} \quad \text{for every } a \in D_x. \tag{8.12}$$

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

LEMMA 8.4. *The solution $\{\tilde{\mathbf{x}}_a\}$ is a feasible solution for SDP relaxation (8.1)–(8.6) without triangle inequalities (8.5) 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,

$$(\mathbf{v}_0 - \tilde{\mathbf{x}}_a)(\mathbf{v}_0 - \tilde{\mathbf{y}}_b) \leq (\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b). \tag{8.13}$$

Observe that

$$\begin{aligned}&(\mathbf{v}_0 - \mathbf{x}_a)(\mathbf{v}_0 - \mathbf{y}_b) - (\mathbf{v}_0 - \tilde{\mathbf{x}}_a)(\mathbf{v}_0 - \tilde{\mathbf{y}}_b) =\\ &\qquad (\mathbf{v}_0 - \tilde{\mathbf{x}}_a)(\tilde{\mathbf{y}}_b - \mathbf{y}_b) + (\tilde{\mathbf{x}}_a - \mathbf{x}_a)(\mathbf{v}_0 - \tilde{\mathbf{y}}_b)\\ &\qquad\qquad + (\tilde{\mathbf{x}}_a - \mathbf{x}_a)(\tilde{\mathbf{y}}_b - \mathbf{y}_b).\end{aligned}$$

We prove that all terms on the right hand side are nonnegative, and thus inequality (8.13) holds. Using the identities (8.12) and $\sum_{a' \in D} \mathbf{x}_{a'} = \mathbf{v}_0$ as well as the inequality $\mathbf{x}_{a'}\mathbf{y}_{b'} \geq 0$ (for all $a', b' \in D$), we get

$$(\mathbf{v}_0 - \tilde{\mathbf{x}}_a)(\tilde{\mathbf{y}}_b - \mathbf{y}_b) = \sum_{\substack{a' \in D \setminus S_{xa}\\ b' \in S_{yb} \setminus \{b\}}} \mathbf{x}_{a'}\mathbf{y}_{b'} \geq 0.$$

Similarly, $(\tilde{\mathbf{x}}_a - \mathbf{x}_a)(\mathbf{v}_0 - \tilde{\mathbf{y}}_b) \geq 0$, and

$$(\tilde{\mathbf{x}}_a - \mathbf{x}_a)(\tilde{\mathbf{y}}_b - \mathbf{y}_b) = \sum_{\substack{a' \in S_{xa} \setminus \{a\}\\ b' \in S_{yb} \setminus \{b\}}} \mathbf{x}_{a'}\mathbf{y}_{b'} \geq 0.$$

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

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

Thus, 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 $\mathcal{C}_1 \setminus \mathcal{C}_{bad}$ 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 et al. for Boolean 2-CSP [16] (when $|D| = 2$, the SDP relaxation used in [16] is equivalent to SDP (8.1)–(8.6) without triangle inequalities (8.5)). We get an assignment of variables in $\mathcal{V}_1$. The weight of constraints in $\mathcal{C}_1 \setminus \mathcal{C}_{bad}$ violated by this assignment is at most $O(\sqrt{\varepsilon})$. Since $w(\mathcal{C}_{bad}) = O(\sqrt{\varepsilon})$, the weight of constraints in $\mathcal{C}_1$ violated by the assignment is at most $O(\sqrt{\varepsilon})$.

8.4 Solving Instance $\mathcal{I}_2$ Instance $\mathcal{I}_2$ is a unique games 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 unique game 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 the algorithm by Charikar et al. [15]; given a $(1-\varepsilon)$-satisfiable instance of Unique Games it finds a solution with the weight of violated constraints at most $O(\sqrt{\varepsilon \log |D|})$. We remark that paper [15] considers only unique game instances. However, in [15], 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 8.2, 8.3, and 8.3 and obtain Theorem 3.1(2).

A Solving instances with exponentially small constraint weights

In this section, we explain how we solve instances with exponentially small constraint weights. As noted in the introduction, we can solve an SDP with an additive error ε' in time $poly(n, \varepsilon')$, where n is the size of the SDP. Therefore, given a $(1-\varepsilon)$-satisfiable instance, we can find an SDP solution of value $(1+2^{-n})\varepsilon$ in time $poly(n, \log 1/\varepsilon)$, which is polynomial in n unless ε is exponentially small. We now outline how we can handle instances with small values of ε.

Consider a $(1-\varepsilon)$ satisfiable instance $\mathcal{I}$ and an optimal combinatorial solution. Let $\tilde{w}$ be the weight of the heaviest constraint that is violated by the solution. Note that $\tilde{w} \le \varepsilon \le m\tilde{w}$ since there are at most m unsatisfied constraints and each of them has weight at most $\tilde{w}$. Since $\tilde{w}$ is the weight of one of the constraints in $\mathcal{C}$, our algorithm may guess the value of $\tilde{w}$ (more precisely, we can run the algorithm for each $\tilde{w} \in \{w_C : C \in \mathcal{C}\}$ and then output the best of the assignments we found).

Given $\tilde{w}$, we perform the following steps to solve the SDP.

- Partition the constraints into two sets
$$\mathcal{C}_{\text{light}} = \{C \in \mathcal{C} : w_C \le \tilde{w}\}$$
and
$$\mathcal{C}_{\text{heavy}} = \{C \in \mathcal{C} : w_C > \tilde{w}\}.$$
Note that the optimal solution satisfies all the constraints in $\mathcal{C}_{\text{heavy}}$.

- Rescale the weights of constraints in $\mathcal{C}_{\text{light}}$ so that they add up to 1; specifically, let $w_C' = w_C / w(\mathcal{C}_{\text{light}})$.

- Write the following SDP relaxation for the problem with a new objective function and extra constraints (A.2): Minimize
$$\text{(A.1)} \qquad \sum_{C=((x,y),R) \in \mathcal{C}_{\text{light}}} w_C' \sum_{(a,b) \notin R} \mathbf{x}_a \mathbf{y}_b$$
subject to
$$\text{(A.2)} \qquad \begin{aligned} \sum_{(a,b)\notin R} \mathbf{x}_a \mathbf{y}_b &= 0 && ((x,y),R) \in \mathcal{C}_{\text{heavy}} \\ \mathbf{x}_a \mathbf{y}_b &\ge 0 && x, y \in V, a, b \in D \\ \mathbf{x}_a \mathbf{x}_b &= 0 && x \in V, a, b \in D, a \ne b \\ \sum_{a \in D} \mathbf{x}_a &= \mathbf{v}_0 && x \in V \\ \|\mathbf{v}_0\| &= 1. \end{aligned}$$
We will refer to this SDP as the auxiliary SDP and to the original SDP (described in Section 4) as the standard SDP. The intuition behind the auxiliary SDP is as follows: its objective function (A.1) measures only the weight of violated constraints in $\mathcal{C}_{\text{light}}$ (w.r.t. weights w_C'); it has additional SDP constraints (A.2) that ensure that all the constraints in $\mathcal{C}_{\text{heavy}}$ are satisfied.

- Observe that the integral SDP solution corresponding to the optimal combinatorial solution is a feasible SDP solution for the auxiliary SDP; namely, it satisfies SDP constraints (A.2) since the combinatorial solution satisfies all the constraints in $\mathcal{C}_{\text{heavy}}$. The value of this SDP solution (w.r.t. to the objective (A.1)) equals the weight of the constraints violated by the optimal solution w.r.t. weights w_C'. Therefore, the optimal SDP value is at most $\tilde{\varepsilon} = \varepsilon / w(\mathcal{C}_{\text{light}})$. Note that $\tilde{\varepsilon} \ge \tilde{w}/(m\tilde{w}) = 1/m$.

- We solve the SDP relaxation with an additive error $2^{-n}/m$ in polynomial-time and obtain an SDP solution $\{\mathbf{x}_a\}_{x\in V, a\in D}$ of value at most $(1+2^{-n})\tilde{\varepsilon}$. Note that $\{\mathbf{x}_a\}_{x\in V, a\in D}$ is a feasible SDP solution to the standard SDP, since the auxiliary SDP has all the SDP constraints from the standard SDP. As a solution to the standard SDP, it has value (4.1) at most

$$\begin{aligned}
&\sum_{C=((x,y),R)\in\mathcal{C}} w_C \sum_{(a,b)\notin R} \mathbf{x}_a\mathbf{y}_b \\
&\quad= \sum_{C=((x,y),R)\in\mathcal{C}_{\text{light}}} w_C \sum_{(a,b)\notin R} \mathbf{x}_a\mathbf{y}_b \\
&\quad+ \sum_{C=((x,y),R)\in\mathcal{C}_{\text{heavy}}} w_C \sum_{(a,b)\notin R} \mathbf{x}_a\mathbf{y}_b \\
&\overset{\text{by (A.2)}}{=} w(\mathcal{C}_{\text{light}}) \sum_{C=((x,y),R)\in\mathcal{C}_{\text{light}}} w'_C \sum_{(a,b)\notin R} \mathbf{x}_a\mathbf{y}_b \\
&\quad+ \sum_{C=((x,y),R)\in\mathcal{C}_{\text{heavy}}} 0 \\
&\quad\leq w(\mathcal{C}_{\text{light}}) \times (1+2^{-n})\tilde{\varepsilon} = (1+2^{-n})\varepsilon.
\end{aligned}$$

- Thus, we obtain an SDP solution to the standard SDP relaxation of value $(1+2^{-n})\varepsilon$.

References

[1] P. Austrin and J. Håstad. On the usefulness of predicates. *ACM Transactions on Computation Theory*, 5(1):1, 2013.

[2] K. Baker and A. Pixley. Polynomial interpolation and the chinese remainder theorem. *Mathematische Zeitschrift*, 143:165–174, 1975.

[3] L. Barto. Finitely related algebras in congruence distributive varieties have near unanimity terms. *Canadian Journal of Mathematics*, 65(1):3–21, 2013.

[4] L. Barto and M. Kozik. Robust satisfiability of constraint satisfaction problems. In *STOC'12*, pages 931–940, 2012.

[5] L. Barto and M. Kozik. Constraint satisfaction problems solvable by local consistency methods. *Journal of the ACM*, 61(1):Article 3, 2014.

[6] L. Barto, M. Kozik, and R. Willard. Near unanimity constraints have bounded pathwidth duality. In *LICS*, pages 125–134, 2012.

[7] W. Bibel. Constraint satisfaction from a deductive viewpoint. *Artificial Intelligence*, 35:401–413, 1988.

[8] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006.

[9] A. Bulatov. Bounded relational width, 2009. manuscript.

[10] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.

[11] A. Bulatov, A. Krokhin, and B. Larose. Dualities for constraint satisfaction problems. In *Complexity of Constraints*, volume 5250 of *LNCS*, pages 93–124. 2008.

[12] C. Carvalho, V. Dalmau, and A. Krokhin. CSP duality and trees of bounded pathwidth. *Theoretical Computer Science*, 411(34-36):3188–3208, 2010.

[13] C. Carvalho, V. Dalmau, and A. Krokhin. Two new homomorphism dualities and lattice operations. *Journal of Logic and Computation*, 21(6):1065–1092, 2011.

[14] S. O. Chan. Approximation resistance from pairwise independent subgroups. In *STOC'13*, pages 447–456, 2013.

[15] M. Charikar, K. Makarychev, and Y. Makarychev. Near-optimal algorithms for unique games. In *STOC'06*, pages 205–214, 2006.

[16] M. Charikar, K. Makarychev, and Y. Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Transactions on Algorithms*, 5(3), 2009.

[17] D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 8. Elsevier, 2006.

[18] M. Cooper, D. Cohen, and P. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65:347–361, 1994.

[19] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. 2001.

[20] V. Dalmau. Linear Datalog and bounded path duality for relational structures. *Logical Methods in Computer Science*, 1(1), 2005. (electronic).

[21] V. Dalmau, M. Kozik, A. Krohkin, K. Makarychev, Y. Makarychev, and J. Opršal. Robust algorithms with polynomial loss for near-unanimity CSPs. Available on *arXiv*, http://arxiv.org/abs/1607.04787, 2016.

[22] V. Dalmau and A. Krokhin. Majority constraints have bounded pathwidth duality. *European Journal of Combinatorics*, 29(4):821–837, 2008.

[23] V. Dalmau and A. Krokhin. Robust satisfiability for CSPs: Hardness and algorithmic results. *ACM Transactions on Computation Theory*, 5(4):Article 15, 2013.

[24] V. Dalmau, A. Krokhin, and R. Manokaran. Towards a characterization of constant-factor approximable Min CSPs. In *SODA'15*, pages 847–857, 2015.

[25] T. Feder, P. Hell, and J. Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42:61–80, 2003.

[26] T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1998.

[27] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[28] V. Guruswami and Y. Zhou. Tight bounds on the approximability of Almost-satisfiable Horn SAT and Exact Hitting Set. *Theory of Computing*, 8:Article 11, 2012.

[29] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001.

[30] J. Håstad. On the NP-hardness of Max-Not-2. *SIAM Journal on Computing*, 43(1):179–193, 2014.

[31] D. Hobby and R. McKenzie. *The Structure of Finite Algebras*, volume 76 of *Contemporary Mathematics*. American Mathematical Society, Providence, R.I., 1988.

[32] P. Jeavons, D. Cohen, and M. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1–2):251–265, 1998.

[33] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[34] S. Khot. On the power of unique 2-prover 1-round games. In *STOC'02*, pages 767–775, 2002.

[35] S. Khot. On the unique games conjecture. In *CCC'10*, pages 99–121, 2010.

[36] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for Max-Cut and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

[37] S. Khot, M. Tulsiani, and P. Worah. A characterization of strong approximation resistance. In *STOC'14*, pages 634–643, 2014.

[38] V. Kolmogorov, A. Krokhin, and M. Rolinek. The complexity of general-valued CSPs. In *FOCS'15*, pages 1246–1258, 2015.

[39] V. Kolmogorov, J. Thapper, and S. Živný. The power of linear programming for general-valued CSPs. *SIAM Journal on Computing*, 44(1):1–36, 2015.

[40] M. Kozik Weak consistency notions for all the CSPs of bounded width. arXiv:1605.00565.

[41] M. Kozik, A. Krokhin, M. Valeriote, and R. Willard. Characterizations of several Maltsev conditions. *Algebra Universalis*, 73(3–4):205–224, 2015.

[42] G. Kun, R. O'Donnell, T. Suguru, Y. Yoshida, and Y. Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In *ITCS'12*, pages 484–495, 2012.

[43] B. Larose, C. Loten, L. Zádori. A polynomial-time algorithm for near-unanimity graphs. *J. Algorithms*, 55(2):177–191, 2005.

[44] B. Larose and P. Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theoretical Computer Science*, 410(18):1629–1647, 2009.

[45] B. Larose, M. Valeriote, and L. Zádori. Omitting types, bounded width and the ability to count. *Internat. J. Algebra Comput.*, 19(5):647–668, 2009.

[46] B. Larose and L. Zádori. Bounded width problems and algebras. *Algebra Universalis*, 56(3–4):439–466, 2007.

[47] M. Maróti and L. Zádori. Reflexive digraphs with near unanimity polymorphisms. *Discrete Mathematics*, 312(15):2316–2328, 2012.

[48] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *STOC'08*, pages 245–254, 2008.

[49] T. Schaefer. The complexity of satisfiability problems. In *STOC'78*, pages 216–226, 1978.

[50] J. Thapper and S. Živný. The complexity of finite-valued CSPs. In *STOC'13*, pages 695–704, 2013.

[51] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[52] U. Zwick. Finding almost-satisfying assignments. In *STOC'98*, pages 551–560, 1998.