Approximation Algorithms for CSPs

Konstantin Makarychev[1] and Yury Makarychev[2]

1 **Microsoft Research**
One Microsoft Way, Redmond, WA 98052, USA
komakary@microsoft.com

2 **Toyota Technological Institute at Chicago**
6045 S Kenwood Ave, Chicago, IL 60637, USA
yury@ttic.edu

1 Introduction

In this survey, we offer an overview of approximation algorithms for constraint satisfaction problems (CSPs) — we describe main results and discuss various techniques used for solving CSPs. We start with recalling standard definitions and introducing the notation.

Constraint Satisfaction Problems. In a constraint satisfaction problem (CSP), we are given a set of variables $x_1, \dots, x_n$ taking values in a domain D of size d, and a set of m constraints (predicates) that depend on the specific problem at hand. Our goal is to find an assignment to the variables that maximizes the number of satisfied constraints. In a weighted CSP, every constraint has a positive weight and our goal is to maximize the total weight of satisfied constraints. All results that we discuss in this survey apply to both unweighted and weighted CSPs. However, for simplicity of exposition, we will only consider the unweighted case. We will say that a CSP is a k-CSP if all constraints have arity at most k.

An instance is $(1-\varepsilon)$-satisfiable if the optimal solution satisfies at least a $(1-\varepsilon)$ fraction of the constraints.

Approximation Algorithms. An approximation algorithm is a (randomized) polynomial-time algorithm that finds an approximate solution. The most common measure of an approximation algorithm's performance is its approximation factor. An algorithm for a maximization problem has an approximation factor $\alpha \le 1$ if it finds a solution of value at least $\alpha\,\mathsf{OPT}$, where OPT is the value of the optimal solution; an algorithm for a minimization problem has an approximation factor $\alpha \ge 1$ if it finds a solution of value at most $\alpha\,\mathsf{OPT}$. We will say that an algorithm is an α-approximation algorithm if it has an approximation factor of α.

Objectives. We consider several objectives for constraint satisfaction problems:

1. Maximize the number of *satisfied* constraints. An α-approximation algorithm for this objective finds a solution that satisfies at least $\alpha\,\mathsf{OPT}$ constraints.
2. Find a solution that satisfies a $1-f(\varepsilon)$ fraction of the constraints given a $(1-\varepsilon)$-satisfiable instance; where f is some function that tends to 0 as $\varepsilon \to 0$ (f should not depend on n).
3. Minimize the number of *unsatisfied* constraints. An α-approximation algorithm for this objective finds a solution that satisfies at least a $(1-\alpha\varepsilon)$ fraction of the constraints given a $(1-\varepsilon)$-satisfiable instance; the approximation factor α may depend on n.

Approximation results for these objectives are often very different. In particular, it makes sense to study objectives (2) and (3) for a given CSP only if there is a polynomial-time algorithm that satisfies all the constraints when all of them are satisfiable (since if an instance is satisfiable, i.e. $\varepsilon = 0$, an algorithm for objective (2) or (3) must satisfy all the constraints). Consider an example — Max 2-Lin(2). This problem is a Boolean CSP of arity 2 with constraints of the form $x_i \oplus x_j = c$ (the problem is a generalization of Max Cut). It can

Dummy Event this volume is based on.
Editor: Editor; pp. 1–40
Dagstuhl Publishing
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany

be solved in linear time if all constraints are satisfiable. For non-satisfiable instances of Max 2-Lin(2), we can get the following results for objectives (1)–(3): obtain a 0.87856-approximation for the maximization variant of the problem [21], satisfy a $1 - O(\sqrt{\varepsilon})$ fraction of the constraints if the optimal solution satisfies a $1 - \varepsilon$ fraction of the constraints [21], and get an $O(\sqrt{\log n})$-approximation for the minimization variant [1]. The first result applies to all instances of Max 2-Lin(2); however, the guarantee it provides for almost satisfiable instances is very weak — even if the instance is completely satisfiable it only guarantees that a 0.87856 fraction of the constraints is satisfied. In contrast, the second result is most interesting for almost satisfiable instances; it guarantees that in such instances the algorithm satisfies almost all constraints. Finally, the third result is meaningful only when the instance is $(1 - c/\log n)$ satisfiable, and, is particularly interesting when $\varepsilon \ll 1/\log^2 n$ — then it gives a much better approximation guarantee than the second result.

Semidefinite Programming. Most state-of-the-art approximation algorithms for constraint satisfaction problems — with the notable exception of the algorithms for Minimum Horn Deletion and Minimum Multiway Cut problems — are based on semidefinite programming. In a semidefinite program (SDP) – written in the vector form – we have

- a set of vector variables $\bar{u}_1, \ldots, \bar{u}_N$;
- an objective function of the form $\sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} \langle \bar{u}_i, \bar{u}_j \rangle$ (where $C = (c_{ij})$ is a real $N \times N$ matrix);
- a set of constraints on $\bar{u}_1, \ldots, \bar{u}_N$, each constraint is of the form $\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} \langle \bar{u}_i, \bar{u}_j \rangle \geq b$ or $\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} \langle \bar{u}_i, \bar{u}_j \rangle = b$ (where $A = (a_{ij})$ is a real $N \times N$ matrix, b is a real number, and $\langle \bar{u}_i, \bar{u}_j \rangle$ denotes the inner/dot product of vectors $\bar{u}_i$ and $\bar{u}_j$).

A feasible solution is an assignment of vectors in $\mathbb{R}^{N-1}$ to variables $\bar{u}_1, \ldots, \bar{u}_N$ that satisfies all the constraints. The goal may be either to find a feasible solution that minimizes or maximizes the objective function. The objective function value for the optimal solution is called the SDP value. It is known that an SDP can be solved with an arbitrary precision in time polynomial in N and the number of constraints.

We note that semidefinite programs can be equivalently written in the matrix form as follows. Let G be the Gram matrix of vectors $\bar{u}_1, \ldots, \bar{u}_N$; that is, let $G = (g_{ij})$ be an $N \times N$ matrix with entries $g_{ij} = \langle \bar{u}_i, \bar{u}_j \rangle$. Write each constraint $\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} \langle \bar{u}_i, \bar{u}_j \rangle = b$ as $\sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} g_{ij} = b$ and the objective function $\sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} \langle \bar{u}_i, \bar{u}_j \rangle$ as $\sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} g_{ij}$. Additionally, require that G be a symmetric positive semidefinite matrix. We get the SDP in the matrix form. Note that every feasible solution $\bar{u}_1, \ldots, \bar{u}_N$ for the SDP in the vector form corresponds to the solution G, with $g_{ij} = \langle \bar{u}_i, \bar{u}_j \rangle$, for the SDP in the matrix form. Further, given a solution $G = (g_{ij})$ for the SDP in the matrix form, the Cholesky decomposition of G provides vectors $\bar{u}_1, \ldots, \bar{u}_N$ such that $g_{ij} = \langle \bar{u}_i, \bar{u}_j \rangle$; vectors $\bar{u}_1, \ldots, \bar{u}_N$ form a solution for the SDP in the vector form.

Writing a semidefinite program in the matrix form, we see that it is a convex program: if G_1 and G_2 are feasible solutions than so is $\alpha G_1 + (1 - \alpha) G_2$ for every $\alpha \in [0, 1]$; the value of the program depends linearly on G.

We will say that a semidefinite program is an SDP relaxation for a constraint satisfaction problem if

- for a minimization problem, the SDP value is at most the value of the CSP;
- for a maximization problem, the SDP value is at least the value of the CSP.

We will construct an SDP relaxation for a CSP using the following approach. For each CSP variable, we introduce one or more vector variables in the SDP relaxation; informally, these vector variables "encode" the CSP variable. Additionally, we may introduce other

vector variables in the SDP relaxation. For every CSP solution $\mathcal{S}$, there is a corresponding feasible SDP solution – the SDP solution that encodes $\mathcal{S}$. This SDP has the same value as the original CSP solution. This guarantees that the SDP is a relaxation: for a minimization problem, the SDP value is at most the value of the SDP solution encoding the optimal CSP solution $\mathcal{S}^*$, which equals the value of $\mathcal{S}^*$; similarly, for a maximization problem, the SDP value is at least the value of $\mathcal{S}^*$. We will refer to a solution that encodes a CSP solution as "intended integral solution". Note that the SDP relaxation may have vector solutions that do not correspond to any CSP solutions. In particular, the optimal SDP solution does not necessarily correspond to any CSP solution.

In order to solve a CSP, we will perform the following steps: write an SDP relaxation for the CSP, find its optimal SDP solution, and then run a so-called *rounding procedure* that transforms the optimal SDP solution to a feasible CSP solution (in general, the rounding procedure does not preserve the value of the solution).

Techniques. We use different SDP relaxations and rounding techniques for different types of CSPs. The key parameters that determine what relaxations and techniques to use are the arity k of the CSP, the domain size d, and the objective.

Boolean 2-CSPs have the simplest SDP relaxations: each variable x_i is encoded by a unit vector $\bar{u}_i$; in the intended integral solution $\bar{u}_i$ is equal to a fixed unit vector $\bar{v}_0 \in S^{n-1}$ if x_i is true, and $\bar{u}_i$ is equal to $-\bar{v}_0$ if x_i is false. Here S^{n-1} denotes the unit sphere in n-dimensional space. The SDP objective function equals the sum of contributions of individual constraints. The contribution of a constraint $\phi(x_i, x_j)$ is

$$\sum_{\alpha,\beta\in\{0,1\}:\phi(\alpha,\beta)} \frac{\langle \bar{v}_0 - (-1)^\alpha \bar{u}_i,\, \bar{v}_0 - (-1)^\beta \bar{u}_j\rangle}{4}$$
$$= \frac{1}{4} \sum_{\alpha,\beta\in\{0,1\}:\phi(\alpha,\beta)} \left(1 + (-1)^{\alpha+\beta}\langle \bar{u}_i, \bar{u}_j\rangle - (-1)^\alpha \langle \bar{u}_i, \bar{v}_0\rangle - (-1)^\beta \langle \bar{u}_j, \bar{v}_0\rangle\right). \quad (1)$$

Here, the summation is over Boolean values α and β that satisfy the predicate $\phi(\alpha, \beta)$; 0 and 1 represent false and true, respectively. For example, the contribution of the constraint $x_i \oplus x_j = 0$ is $(\langle \bar{v}_0 - \bar{u}_i, \bar{v}_0 - \bar{u}_j\rangle + \langle \bar{v}_0 + \bar{u}_i, \bar{v}_0 + \bar{u}_j\rangle)/4 = (\|\bar{v}_0\|^2 + \langle \bar{u}_i, \bar{u}_j\rangle)/2 = (1 + \langle \bar{u}_i, \bar{u}_j\rangle)/2$, the contribution of $x_i \vee x_j$ is $(3 + \langle \bar{u}_i + \bar{u}_j, \bar{v}_0\rangle - \langle \bar{u}_i, \bar{u}_j\rangle)/4$. The SDP relaxation has constraints $\|\bar{u}_i\|^2 \equiv \langle u_i, u_i\rangle = 1$ and, possibly, some additional constraints that depend on the CSP.

Let us consider how an SDP algorithm works at a high level. We solve the SDP relaxation and find a (nearly) optimal SDP solution $\{\bar{u}_i\}$. The SDP solution may be very different from the intended solution; in particular, vectors $\{\bar{u}_i\}$ do not have to be equal or close to vectors $\bar{v}_0$ or $-\bar{v}_0$. We use a randomized rounding procedure to transform the set of vectors $\bar{u}_i$ to a Boolean assignment for variables x_i. To ensure that the value of the obtained assignment is large, we want to transform near-by vectors to the same value with a high probability and antipodal vectors to opposite values. For some problems, we do the rounding in one step; for other problems, we use an iterative procedure to do the rounding. In the former case, it is instructive to think of the rounding procedure as consisting of two actions:

- Generate a random partition $(A, S^{n-1} \setminus A)$ of the unit sphere S^{n-1} into two pieces, which is symmetric about the origin (that is, $A = -(S^{n-1} \setminus A)$).
- Assign $x_i = 1$ if $x \in A$, and $x_i = 0$ if $x \notin A$.

Usually, the distribution of random partitions of S^{n-1} does not depend on the SDP solution, except that it may depend on the value of the SDP solution. We note that, in this case, we can always assume that the distribution of random partitions is invariant under the action of

the subgroup $\{T \in O_n : Tv_0 = v_0\}$ of orthogonal transformations that map v_0 to v_0.[1]

To prove an approximation guarantee for this algorithm, we need to lower bound the probability that each constraint $\phi(x_i, x_j)$ is satisfied in terms of its SDP contribution. To this end, we only have to analyze how the random partition divides vectors $\bar{u}_i$ and $\bar{u}_j$ depending on the angles between them and between them and $\bar{v}_0$.

SDP relaxations for non-Boolean 2-CSPs are more complex. Consider a 2-CSP with domain size $d > 2$; let $D = \{1, \ldots, d\}$ be its domain. Now, to encode a variable x_i, we introduce d SDP vectors $\bar{u}_{i1}, \ldots, \bar{u}_{id}$. In the intended solution, $\bar{u}_{ij} = \bar{v}_0$ if $x_i = j$ and $\bar{u}_{ij} = 0$ otherwise. The SDP contribution of the constraint $\phi(x_i, x_j)$ equals $\sum_{\alpha,\beta \in D:\phi(\alpha,\beta)} \langle \bar{u}_{i\alpha}, \bar{u}_{j\beta} \rangle$. The SDP has constraints that require that $\sum_{j=1}^{d} \bar{u}_{ij} = \bar{v}_0$ (this constraint can be written equivalently as $\left\| \sum_{j=1}^{d} \bar{u}_{ij} - \bar{v}_0 \right\|^2 = 0$), $\sum_{j=1}^{d} \|\bar{u}_{ij}\|^2 = 1$, all vectors $u_{i1}, \ldots, u_{id}$ are mutually orthogonal, as well as additional constraints that depend on the CSP. Informally, we can interpret $\|u_{ij}\|^2$ as the desired probability of the event $x_i = j$ and $\langle u_{i_1 j_1}, u_{i_2 j_2} \rangle$ as the desired probability of the event $x_{i_1} = j_1$ and $x_{i_2} = j_2$. Then the SDP constraints say that the sum of the probabilities of the events $x_i = j$ over all j is equal to 1, and events $x_i = j_1$ and $x_i = j_2$ are mutually exclusive.

Rounding an SDP solution for a non-Boolean 2-CSP is considerably more challenging than rounding an SDP solution for a Boolean 2-CSP. Now for each variable x_i, we want to choose exactly one vector $\bar{u}_{ij}$ among d vectors $\bar{u}_{i1}, \ldots, \bar{u}_{id}$ and assign $x_i = j$. Note that we cannot simply use the same approach as before — choose a random subset A of Euclidean space, and let $x_i = j$ if $\bar{u}_{ij} \in A$, because we cannot choose a random subset A so that exactly one of any k orthogonal vectors belong to A (in contrast, it is easy to find a subset A of S^{n-1} so that exactly one of the vectors $\bar{u}$ and $-\bar{u}$ is in A). Consequently, if we try to implement such a scheme, we will sometimes assign no value or more than one value to x_j. One approach to fix this problem is to use an iterative rounding procedure:

- Find a random subset A such that the probability that two given orthogonal vectors belong to it is sufficiently small (namely, it should it be at most $1/d^c$ for some $c > 1$).
- Assign $x_i = j$, if $\bar{u}_{ij} \in A$ and there is no $j' \neq j$ such that $\bar{u}_{ij'} \in A$. Get a partial assignment to variables x_i.
- If there are unassigned variables, repeat this procedure. Do not change the values of the already assigned variables.

We note that some algorithms do not use this approach and assign values to all variables in one step (see e.g. [14]). However, as we will see in Section 3, this approach allows us to considerably simplify the algorithms' analysis; loosely speaking, to lower bound the probability that $\phi(x_i, x_j)$ is satisfied, we only need to lower bound the probabilities $\Pr(\bar{u}_{i_2 j_2} \in A \,|\, \bar{u}_{i_1 j_1} \in A)$ and $\Pr(\bar{u}_{i_1 j_1} \in A \,|\, \bar{u}_{i_2 j_2} \in A)$ for $j_1, j_2 \in D$ satisfying the constraint $\phi(j_1, j_2)$ (both probabilities are over the random choice of A). When we do that, we can restrict our attention to vectors $\bar{u}_{i_1 j_1}$ and $\bar{u}_{i_2 j_2}$, and not have to analyze all possible spatial configurations of vectors $\bar{u}_{i_1 1}, \ldots, \bar{u}_{i_1 d}, \bar{u}_{j_1 1}, \ldots, \bar{u}_{j_1 d}$. Nevertheless, the analysis is still more complicated than that for Boolean 2-CSPs. Particularly, it is very important to properly handle both directions and lengths of vectors.

---

[1] Indeed, any orthogonal transformation $T \in O_n$ such that $Tv_0 = v_0$ maps a feasible SDP solution $\{\bar{u}_i\}$ to a feasible SDP solution $\{T\bar{u}_i\}$ of the same value. So instead of applying the rounding procedure to $\{\bar{u}_i\}$, we can sample T uniformly at random from $G = \{T \in O_n : Tv_0 = v_0\}$ and apply the rounding procedure to a transformed solution $\{T\bar{u}_i\}$. Equivalently, we can randomly sample T and a random partition $(A, S^{n-1} \setminus A)$, let $A' = T^{-1}A$, and assign $x_i = 1$ if $x \in A'$, and $x_i = 0$ if $x \notin A'$. Note that the distribution of A' is invariant under the action of G. Thus, we can replace the distribution of partitions A with the G-invariant distribution of partitions A' in our approximation algorithm.

problem	**constraints** (z_i **is either** x_i **or** $\bar{x}_i$)	**approx. factor**	**optimal? upper bound**
Max Cut	$x_i \neq x_j$	0.87856 [21]	yes [33]
Max 2-Lin(2)	$x_i \oplus x_j = c_{ij}$		
Max 2-SAT	$z_i \vee z_j$	0.94016 [39]	yes [6]
Max Di-Cut	$\bar{x}_i \wedge x_j$		0.87856 [33]
Max 2-And	$z_i \wedge z_j$	0.87401 [39]	0.87435 [6]
Any Boolean 2-CSP	Boolean 2-CSP		0.87435 [6]
Max 3-SAT	$\bigvee_{j=1}^{t} z_{i_j}$ $(t \leq 3)$	7/8 [31, 54]	yes [26]
Max E3-SAT	$\bigvee_{j=1}^{3} z_{i_j}$		
Any Boolean k-CSP	Boolean k-CSP	$\frac{(0.62661-o(1))k}{2^k}$ [43]	$\frac{(1+o(1))k}{2^k}$ [7, 13]
Max k-And	$z_{i_1} \wedge \cdots \wedge z_{i_k}$		
Max SAT	$\bigvee_{j=1}^{t} z_{i_j}$	0.7968 [8]	7/8 [26]
		conj. 0.8434 [8]	
Max Ek-SAT $(k \geq 3)$	$\bigvee_{j=1}^{k} z_{i_j}$	$1 - 1/2^k$	yes [26]
Max k-All-Equal	$z_{i_1} = \cdots = z_{i_k}$	$\frac{0.88007\,k}{2^k}$ [15]	$\frac{(2+o(1))k}{2^k}$ [7, 13]
Max k-NAE-SAT	$\overline{z_{i_1} = \cdots = z_{i_t}}$ $(t \leq k)$	0.7499 [52]	0.87856 [33]
		conj. 0.8279 [8]	
Max k-Lin(2) $(k \geq 3)$	$z_{i_1} \oplus \cdots \oplus z_{i_k} = 0$	1/2	yes [26]

Figure 1 List of known positive and negative results for Boolean CSPs with the maximization objective. Some hardness results assume UGC and some assume only that $P \neq NP$.

Rounding SDPs for most CSPs of arity $k > 2$ — and especially non-Boolean CSPs of arity $k > 2$ — poses additional challenges. The standard SDP relaxation for CSPs of arity $k > 2$ is somewhat similar to that for non-Boolean 2-CSPs; the key difference is that for each constraint $\phi(x_{i_1}, \ldots, x_{i_k})$ and each satisfying assignment $x_{i_1} = j_1, \ldots, x_{i_k} = j_k$ for this constraint, we have an additional SDP vector variable $\bar{v}_{(i_1,j_1),\ldots,(i_k,j_k)}$. In the intended solution, $\bar{v}_{(i_1,j_1),\ldots,(i_k,j_k)} = \bar{v}_0$ if $x_{i_1} = j_1, \ldots, x_{i_k} = j_k$, and $\bar{v}_{(i_1,j_1),\ldots,(i_k,j_k)} = 0$, otherwise. The SDP objective function equals the sum of $\|\bar{v}_{(i_1,j_1),\ldots,(i_k,j_k)}\|^2$ over all variables $\bar{v}_{\ldots}$. There are additional SDP constraints of the form $\langle v_{(i_1,j_1),\ldots,(i_k,j_k)}, u_{i_t,j_t}\rangle = \|v_{(i_1,j_1),\ldots,(i_k,j_k)}\|^2$ and $\langle v_{(i_1,j_1),\ldots,(i_k,j_k)}, u_{i_t,j'}\rangle = 0$ if $j' \neq j_t$. The main challenge is that for such problems as Max k-And, to lower bound the probability that a constraint $\phi(x_{i_1}, \ldots, x_{i_k})$ is satisfied, we have to analyze the spatial configuration of all dk vectors $x_{i_1 1}, \ldots, x_{i_1 d}, \ldots, x_{i_k 1}, \ldots, x_{i_k d}$ and vector $\bar{v}_{(i_1,j_1),\ldots,(i_k,j_k)}$.

Metric Embedding Techniques. Low-distortion metric embedding techniques are among the most powerful and widely used in combinatorial optimization. Not surprisingly, they are also employed for solving certain CSPs: Min UnCut, Min 2CNF Deletion, and Unique Games (in all these problems, the objective is to minimize the number of unsatisfied constraints). However, we do not describe any algorithms that use metric embeddings in this survey; we refer the reader to papers [1, 16].

1.1 Overview of Known Results for CSPs

In this section, we give an overview of known approximation results for constraint satisfaction problems.

Boolean CSPs. First, we discuss the results for Boolean CSPs with the maximization objective (objective (1) in our list). The results are summarized in Figure 1. When we describe a CSP, we write z_i to denote a literal x_i or $\bar{x}_i$. The most basic Boolean Max 2-CSP problem is Max Cut. In this problem, each constraint is of the form $x_i \neq x_j$, or, equivalently, $x_i \oplus x_j = 1$. Goemans and Williamson designed a 0.87856 approximation algorithm for the problem [21]. Later, Khot, Kindler, Mossel, and O'Donnell showed that this algorithm is optimal assuming the Unique Games Conjecture (UGC)[33]. The best unconditional hardness result was obtained by Håstad [26], who showed that it is NP-hard to obtain a better than $16/17 \approx 0.94117$ approximation (i.e., for every constant $\delta > 0$, it is impossible to get a $(16/17 - \delta)$ approximation in polynomial time if $P \neq NP$). All these results for Max Cut also apply to a more general Max 2-Lin(2) problem, a Boolean 2-CSP with constraints of the form $x_i \oplus x_j = c$ (where $c \in \{0, 1\}$).

Lewin, Livnat, and Zwick gave a 0.94016-approximation algorithm for Max 2-SAT, a problem with disjunctive constraints of the form $z_i \vee z_j$ [39]; Austrin proved that this algorithm is optimal assuming UGC [6]. Lewin et al. also designed a 0.87401-approximation algorithm for Max 2-And, a problem with conjunctive constraints of the form $z_i \wedge z_j$. This problem is the most general maximization Boolean 2-CSP — there is an approximation-preserving reduction from any Max Boolean 2-CSP to Max 2-And (thus, if there is an α-approximation for Max 2-And, then there is an α-approximation for any Boolean 2-CSP). Therefore, the algorithm by Lewin et al. gives a 0.87401 approximation for any Boolean 2-CSP. The approximation factor of 0.87401 is not known to be optimal — the best upper bound, due to Austrin [6], is 0.87435; note that the gap between the lower and upper bounds is less than 0.0004.

Now consider CSPs of greater arities. In Max 3-SAT, each constraint is a disjunction of at most 3 literals: $z_{i_1} \wedge \cdots \wedge z_{i_t}$ $(t \leq 3)$; in Max E3-SAT, each constraint is a disjunction of exactly 3 literals. There is a trivial 7/8-approximation algorithm for Max 3E-SAT — simply choose each x_i uniformly at random from $\{0, 1\}$ (the algorithm can be easily derandomized using the method of conditional expectations). Max 3-SAT is more difficult — Karloff and Zwick showed how to get a 7/8-approximation if a certain conjecture is true [31]; then, Zwick gave a computer-assisted proof that there is indeed a 7/8-approximation algorithm for the problem [54]. Håstad showed that Max E3-SAT is approximation resistant [26] and, thus, Max E3-SAT and Max 3-SAT do not admit a better than 7/8 approximation if $P \neq NP$.

Avidor, Berkovitch, and Zwick [8] studied the general Max SAT problem, in which each constraint is a disjunction of an arbitrary number of literals. They showed how to get a 0.7968 approximation for the problem; additionally, they gave an algorithm that gets a 0.8434 approximation if a certain conjecture is true. No hardness results have been proved specifically for Max SAT; however, Håstad's 7/8 hardness for Max E3-SAT also applies to Max SAT. Finally, we note that Max Ek-SAT, the problem in which each constraint is a disjunction of exactly k literals, is considerably simpler than Max SAT. The random assignment algorithm gives a $1 - 1/2^k$ approximation; Håstad showed that there is no better approximation algorithm for the problem when $k \geq 3$ [26].

Consider an arbitrary Boolean k-CSP with the maximization objective. There is an approximation-preserving reduction from the problem to Max k-And (the problem in

which every constraint is a conjunction of k literals[2]). The algorithm by Makarychev and Makarychev [43] gives a $(0.62661 - o(1))k/2^k$ approximation for Max k-And and thus for any Max Boolean 2-CSP (the little $o(1)$ term tends to 0 as $k \to \infty$). Austrin and Mossel proved a $(1 + o(1))k/2^k$ hardness of approximation if UGC [7] is true. Later, Chan [13] showed the hardness of $2k/2^k$ if $P \neq NP$; further, he proved the hardness of $(1 + o(1))k/2^k$ for infinitely many k if $P \neq NP$. Note that the lower and upper bounds differ only by a constant factor; we conjecture that the algorithm in [43] actually gets a $(1 - o(1))k/2^k$ approximation. We note that the first asymptotically optimal, up to constant factors, upper and lower bounds for the problem were obtained by Samorodnitsky and Trevisan [49] and by Charikar, Makarychev, Makarychev [15], respectively. The algorithm in [15] gives a $0.44003k/2^k$ approximation for all values of k.

In Figure 1, we also summarize known results for several other Max Boolean CSPs: Max Di-Cut, Max k-All-Equal, Max k-NAE-Equal, and Max k-Lin(2).

Now, we briefly describe results for almost satisfying instances of Boolean CSPs (with objectives (2) and (3) from our list). The results are shown in Figure 2. The algorithm by Goemans and Williamson for Max Cut satisfies a $1 - O(\sqrt{\varepsilon})$ fraction of the constraints if the optimal solution satisfies a $1 - \varepsilon$ fraction of the constraints [21]. Charikar, Makarychev, and Makarychev [15] gave an algorithm for all Boolean 2-CSPs with the same approximation guarantee of $1 - O(\sqrt{\varepsilon})$. Khot, Kindler, Mossel, and O'Donnell [33] showed that these results are asymptotically optimal if UGC is true.

Agarwal, Charikar, Makarychev, and Makarychev designed an $O(\sqrt{\log n})$ approximation algorithms for Min Uncut and Min 2-CNF Deletion, the minimization versions of Max Cut and Max 2-SAT, respectively. The algorithm for Min 2-CNF Deletion gives an $O(\sqrt{\log n})$ approximation to arbitrary minimization Boolean 2-CSPs. The $(1 - O(\sqrt{\varepsilon}))$-hardness result by Khot et al. implies that there is no constant factor approximation for Min Uncut, Min 2-CNF Deletion, and, in general, Min Boolean 2-CSPs if UGC is true. Chlebík and Chlebíková proved an unconditional $8\sqrt{5} - 15 \approx 2.88854$ NP-hardness of approximation for Min 2-CNF Deletion [17]. Håstad, Huang, Manokaran, O'Donnell, and Wright [27] proved an unconditional $11/8 = 1.375$ NP-Hardness of approximation for Min Uncut.

Finally, we describe results for Max Horn SAT (Min Horn Deletion). Recall that a Horn clause is a disjunction of literals with at most one positive (non-negated) literal.[3] There is no approximation algorithm specifically for Horn SAT with the maximization objective; the approximation algorithm by Avidor et al. for arbitrary SAT instances also gives a 0.7968 approximation for Max Horn SAT [8]. Zwick [53] designed an algorithm for $(1 - \varepsilon)$ satisfiable instances of Max Horn SAT (the problem is also called Min Horn Deletion); the algorithm satisfies at least a $(1 - 8 \log\log \frac{1}{\varepsilon} / \log \frac{1}{\varepsilon})$ fraction of the constraints. Guruswami and Zhou proved an almost matching UGC-hardness result of $(1 - \Omega(1/\log \frac{1}{\varepsilon}))$ (note that the upper and lower bounds differ by a $\log\log \frac{1}{\varepsilon}$ factor) [25]. They also presented an algorithm that satisfies a $(1 - 2\varepsilon)$ fraction of the constraints given a $(1 - \varepsilon)$-satisfiable instance of Max Horn 2-SAT and showed that this result is optimal (if UGC is true) [25].

Non-Boolean Max k-CSP. In Max k-CSP(d), an instance is a CSP with arbitrary constraints of arity k over a domain of size d. There is an approximation preserving reduction

[2] The variant of the problem, in which each constraint is a conjunction of *at most* k literals, is equivalent to the variant, in which each constraint is a conjunction of *exactly* k literals.

[3] Some authors define a Horn clause as a disjunction of literals with at most one *negated* literal (see e.g. [53]). The two definitions are different; however, by negating all literals, an instance of one problem can be transformed to an instance of the other problem of the same value.

problem	objective	satisfied constraints	optimal? hardness result
Max Cut Max 2-SAT Any Boolean 2-CSP	(2)	$1 - O(\sqrt{\varepsilon})$ [21, 15]	yes [33]
Max Cut Max 2-SAT Any Boolean 2-CSP	(3)	$1 - O(\varepsilon\sqrt{\log n})$ [1]	No $O(1)$ approx. [33]
Max Horn SAT	(2)	$1 - 8\log\log\frac{1}{\varepsilon}/\log\frac{1}{\varepsilon}$ [53]	$1 - \Omega\big(1/\log\frac{1}{\varepsilon}\big)$ [25]
Max Horn 2-SAT	(2) and (3)	$1 - 2\varepsilon$ [25]	yes [25]

Figure 2 List of known results for almost satisfiable instances of Boolean CSPs. The table shows what fraction of the constraints we can satisfy if the optimal solution satisfies a $(1 - \varepsilon)$ fraction of the constraints. Note that the minimization versions of Max Cut, Max 2-SAT, and Max Horn SAT are known as Min Uncut, Min 2-CNF Deletion, and Min Horn Deletion, respectively.

from Max k-CSP(d) to the CSP with constraints of the form $(x_{i_1} = j_1) \wedge \cdots \wedge (x_{i_k} = j_k)$. Makarychev and Makarychev designed an $\Omega(dk/d^k)$ approximation algorithm[4] for the case when $k \geq \Omega(\log d)$. Most recently, Manurangsi, Nakkiran, and Trevisan [46] gave an $\Omega(d \log d/d^k)$ approximation algorithm for $k \leq O(\log d)$ (see also [35]). Relying on the work of Austrin and Mossel [7], Håstad proved a hardness of $\Omega(kd/d^k)$ for $k \geq d$, assuming UGC [43]. Later, Chan proved that this hardness result holds if $P \neq NP$ [13]. His results also imply an $O(d^2/d^k)$ hardness of approximation for $k < d$ and an $O(\log d/\sqrt{d})$ hardness for $k = 2$. Additionally, Manurangsi et al. [46] showed an $\frac{2^{O(k \log k)} d\,(\log d)^{k/2}}{d^k}$-hardness of approximation, assuming UGC (this upper bound on the approximation factor is better than that by Chan when $k \ll \frac{\log d}{\log\log d}$). To summarize, the best known approximation factor for the problem is $\Omega(d \max(k, \log d)/d^k)$; it is known to be optimal up to a constant factor when $k \geq d$, if $P \neq NP$, and also when $k = 2$, if UGC holds.

Other CSPs. We describe known results for Unique Games in Section 3 and results for Minimum Multiway Cut in Section 5.

Universal Algorithm for CSPs. In Section 6, we discuss two very important general results on approximability of CSPs: the result by Raghavendra [47] that shows that semidefinite programming gives the best possible approximation for many CSPs and the universal approximation algorithm for CSPs by Raghavendra and Steurer [48].

1.2 Organization

In Section 2, we describe the algorithm by Goemans–Williamson for Max Cut and discuss algorithms for other Boolean 2-CSPs. In Section 3, we describe known results for Unique Games and give an approximation algorithm for the problem, as well as present the framework of orthogonal separators. Then in Section 4, we discuss techniques for solving CSPs of arities $k > 2$. In Section 5, we discuss the Minimum Multiway Cut problem and present the algorithm by Călinescu, Karloff, and Rabani [12]. This is the only algorithm based

[4] We write the approximation factor as dk/d^k and not as k/d^{k-1}, because it is easier to compare it to the approximation factor $1/d^k$ of the random assignment algorithm, when it is written in this form.

on linear programming that we give in this survey; all other algorithms are based on semidefinite programming. Finally, in Section 6, we discuss the results by Raghavendra [47] and Raghavendra and Steurer [48] and describe and analyze the universal rounding algorithm for (generalized) 2-CSPs with nonnegative predicates [48]. We conclude the paper with a list of open problems.

2 Boolean CSPs or arity 2: Max Cut and Max 2-SAT

In this section, we describe the Goemans–Williamson approximation algorithm [21] for Max Cut and discuss approximation algorithms for other Boolean 2-CSPs.

Max Cut problem may be stated as a graph partitioning or constraint satisfaction problem. In the graph partitioning formulation, we are given a graph $G = (V, E)$, and our goal is to find a cut $(S, \bar{S})$ so as to maximize the number of cut edges. In the CSP formulation, we are given a set of variables $x_1, \dots, x_n$ and a set of constraints of the form $x_i \neq x_j$; our goal is to find a Boolean assignment that maximizes the number of satisfied constraints. There is a simple correspondence between the two formulations: vertices correspond to variables, edges correspond to the constraints, and a solution $(S, \bar{S})$ corresponds to a CSP solution that assigns 1 (true) to vertices in S and 0 (false) to vertices in $\bar{S}$. Below, we consider the CSP formulation of the problem. We note that all results we describe in this section also apply to a more general Max 2-Lin(2) problem.

▶ **Theorem 1** ([21]). *There exists a randomized polynomial-time approximation algorithm for Max Cut that finds a solution of value at least $\alpha_{GW}\mathsf{OPT}$, in expectation, where $\alpha_{GW} \approx 0.87856$.*

Proof. We write the following standard SDP relaxation for Max Cut.

$$\text{maximize } \frac{1}{4} \sum_{\text{constraint } x_i \neq x_j} \|\bar{u}_i - \bar{u}_j\|^2$$

subject to

$$\|\bar{u}_i\|^2 = 1 \qquad \text{for every } i \in \{1, \dots, n\}.$$

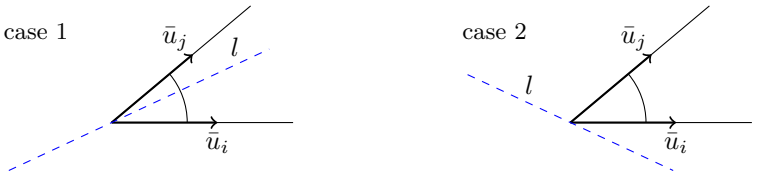
There is an SDP vector variable $\bar{u}_i$ for each CSP variable x_i. The only constraint in the SDP requires that all vectors $\bar{u}_i$ be unit vectors.

Let us verify this is indeed a relaxation; that is, the optimal value of the SDP is at least OPT. To this end, consider an optimal solution $x_i = \hat{x}_i$. Now, pick an arbitrary unit vector $\bar{v}_0$, and define a feasible SDP solution $\bar{u}_i = \bar{v}_0$ if $x_i = 1$, and $\bar{u}_i = -\bar{v}_0$ if $x_i = 0$. Observe that $\frac{1}{4}\|\bar{u}_i - \bar{u}_j\|^2 = 1$ if $x_i \neq x_j$, and $\frac{1}{4}\|\bar{u}_i - \bar{u}_j\|^2 = 0$, otherwise. Thus, the value of this SDP solution equals OPT. Therefore, the value of the optimal SDP solution is at least OPT. We denote the value of the SDP solution by SDP.

The Goemans–Williamson approximation algorithm solves the SDP relaxation and finds an optimal solution $\{\bar{u}_i\}$. Now, it chooses a random hyperplane H passing through the origin. The hyperplane partitions space into two half-spaces A and $\bar{A}$ (we arbitrarily choose which of the half-spaces is A and which is $\bar{A}$). The algorithm returns the following solution:

$$x_i = \begin{cases} 1, & \text{if } \bar{u}_i \in A, \\ 0, & \text{if } \bar{u}_i \in \bar{A}. \end{cases}$$

Let us analyze this algorithm. Consider a constraint $x_i \neq x_j$. We lower bound the probability that it is satisfied. Consider the two dimensional plane P passing through $\bar{u}_i$ and $\bar{u}_j$. Note that the intersection between P and the random hyperplane H is a random line l in P, passing through the origin. Consider line l and the angle formed by vectors $\bar{u}_i$ and $\bar{u}_j$; note that l goes through the vertex of the angle.



There are two cases: (1) P separates the sides of the angle and (2) it does not. In the former case, $\bar{u}_i$ and $\bar{u}_j$ lie in different half-spaces w. r. t. H; that is, one of them is in A and the other is in $\bar{A}$. The algorithm assigns different values to x_i and x_j, and thus satisfies the constraint $x_i \neq x_j$. In the latter case, $\bar{u}_i$ and $\bar{u}_j$ lie in the same half-space w. r. t. H; the algorithm assigns the same value to x_i and x_j, and thus violates the constraint $x_i \neq x_j$. We conclude that the probability that the constraint $x_i \neq x_j$ is satisfied equals the probability that l goes between $\bar{u}_i$ and $\bar{u}_j$. This probability equals the angle between $\bar{u}_i$ and $\bar{u}_j$ divided by π: $\arccos\langle \bar{u}_i, \bar{u}_j\rangle/\pi$. We compare this probability with the SDP contribution of the constraint $x_i \neq x_j$: $\frac{1}{4}\|\bar{u}_i - \bar{u}_j\|^2 = \frac{1-\langle \bar{u}_i, \bar{u}_j\rangle}{2}$.

$$\Pr(x_i \neq x_j) \Big/ \frac{1-\langle \bar{u}_i, \bar{u}_j\rangle}{2} = \frac{2}{\pi}\frac{\arccos\langle \bar{u}_i, \bar{u}_j\rangle}{1-\langle \bar{u}_i, \bar{u}_j\rangle} \geq \min_{x\in[-1,1]} \frac{2}{\pi}\frac{\arccos x}{1-x} \equiv \alpha_{GW} \geq 0.87856.$$

Here, α_{GW} is the minimum of the function $\frac{2}{\pi}\frac{\arccos x}{1-x}$ on $[-1,1]$. Numerically, it is greater than and approximately equal to 0.87856.

We conclude that, in expectation, the algorithm satisfies at least

$$\sum_{\text{constraint } x_i\neq x_j} \Pr(x_i \neq x_j) \geq \frac{\alpha_{GW}}{4} \sum_{\text{constraint } x_i\neq x_j} \|\bar{u}_i - \bar{u}_j\|^2 = \alpha_{GW}\mathsf{SDP} \geq \alpha_{GW}\,\mathsf{OPT}$$

constraints, as required. Note that if we run the algorithm sufficiently many times and output the best of the solutions we find, we get at least an $(\alpha_{GW} - \delta)$ approximation with high probability (for a polynomially small δ). ◀

▶ **Theorem 2** (Goemans and Williamson [21]). *Given a $(1-\varepsilon)$-satisfiable instance of Max Cut, the Goemans–Williamson algorithm finds a solution of value at least $1-\sqrt{\varepsilon}$, in expectation.*

Proof. Let m be the number of the constraints. As we showed in the proof of Theorem 1, in expectation, the fraction of the constraints satisfied by the Goemans–Williamson algorithm is

$$\frac{1}{m}\sum_{\text{constraint } x_i\neq x_j} \Pr(x_i \neq x_j) = \frac{1}{m}\sum_{\text{constraint } x_i\neq x_j} \frac{\arccos\langle \bar{u}_i, \bar{u}_j\rangle}{\pi}.$$

We now use the inequality (see Figure 3)

$$\frac{\arccos t}{\pi} \geq 1 - \sqrt{\frac{1+t}{2}}$$

for t in $[-1,1]$. This inequality holds because function $g(t) = \frac{\arccos t}{\pi} + \sqrt{\frac{1+t}{2}} - 1$ is concave; $g(-1) = g(1) = 0$ and, consequently, $g(t) \geq 0$ for $t \in [-1,1]$. Thus, the expected fraction of

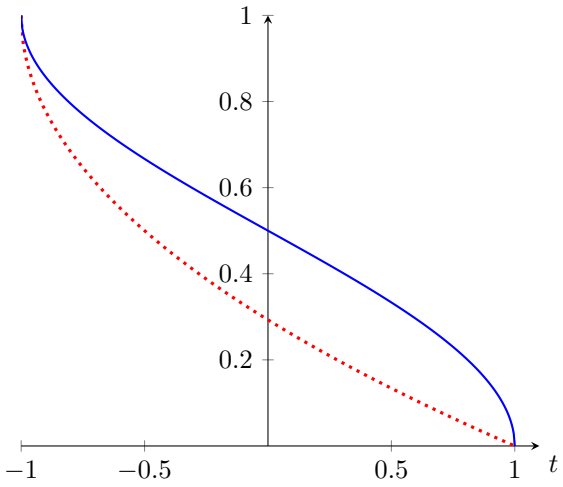


Figure 3 Plots of functions $t \mapsto \arccos(t)/\pi$ (above) and $t \mapsto 1 - \sqrt{(1+t)/2}$ (below).

satisfied constraints is at least

$$\frac{1}{m} \sum_{\text{constraint } x_i \neq x_j} \frac{\arccos\langle \bar{u}_i, \bar{u}_j \rangle}{\pi} \geq \frac{1}{m} \sum_{\text{constraint } x_i \neq x_j} \left(1 - \sqrt{\frac{1 + \langle \bar{u}_i, \bar{u}_j \rangle}{2}}\right).$$

By Jensen's inequality for the concave function $t \mapsto \sqrt{t}$, we have

$$\frac{1}{m} \sum_{\text{constraint } x_i \neq x_j} \left(1 - \sqrt{\frac{1 + \langle \bar{u}_i, \bar{u}_j \rangle}{2}}\right) \geq 1 - \sqrt{\frac{1}{m} \sum_{\text{constraint } x_i \neq x_j} \frac{1 + \langle \bar{u}_i, \bar{u}_j \rangle}{2}}.$$

Now, observe that the expression under the root on the right hand side of the inequality exactly equals $1 - SDP/m$, where SDP is the SDP value of the optimal solution $\{\bar{u}_j\}$. Hence, the expected fraction of satisfied constraints is at least $1 - \sqrt{1 - SDP/m}$. Since the given Max Cut instance is $1 - \varepsilon$ satisfiable, we have $SDP/m \geq OPT/m \geq 1 - \varepsilon$. We conclude that the expected fraction of satisfied constraints by the Goemans–Williamson algorithm is at least $1 - \sqrt{\varepsilon}$.

Note that by running this algorithm many times, we can find a solution satisfying a $1 - O(\sqrt{\varepsilon})$ fraction of the constraints with high probability. ◀

Approximation algorithms for other Boolean 2-CSPs are more complex. Consider the Max 2-SAT problem, in which constraints are of the form $z_i \vee z_j$ (where z_i and z_j are literals). Notationally, it is convenient to introduce variables $x_{-1}, \ldots, x_{-n}$ for the negated literals $\bar{x}_1, \ldots, \bar{x}_n$; that is, let $x_{-i} = \bar{x}_i$. Then, every constraint can be written as $x_i \vee x_j$, where $i, j \in \{\pm 1, \ldots, \pm n\}$. We write an SDP relaxation for the problem. We have SDP variables

$\bar{u}_{\pm 1}, \ldots, \bar{u}_{\pm n}$ for CSP variables $x_{\pm 1}, \ldots, x_{\pm n}$. We require that $\bar{u}_i = -\bar{u}_{-i}$.

$$\text{maximize } \frac{1}{4} \sum_{\text{constraint } x_i \vee x_j} (3 + \langle \bar{u}_i + \bar{u}_j, \bar{v}_0 \rangle - \langle \bar{u}_i, \bar{u}_j \rangle)$$

subject to

$$\frac{3 + \langle \bar{u}_i + \bar{u}_j, \bar{v}_0 \rangle - \langle \bar{u}_i, \bar{u}_j \rangle}{4} \leq 1 \qquad \text{for every } i \in \{\pm 1, \ldots, \pm n\}$$

$$\bar{u}_i = -\bar{u}_{-i} \qquad \text{for every } i \in \{\pm 1, \ldots, \pm n\}$$

$$\|\bar{u}_i\|^2 = \|v_0\|^2 = 1 \qquad \text{for every } i \in \{1, \ldots, \pm n\}$$

Note that we need an extra variable $\bar{v}_0$ in the relaxation for Max 2-SAT (which was absent in the relaxation for Max Cut). Variable $\bar{v}_0$ represents true; accordingly, $-\bar{v}_0$ represents false. In the intended SDP solution corresponding to a CSP solution, $\bar{v}_0$ is an arbitrary unit vector, and $\bar{u}_i = \bar{v}_0$ if $x_i = 1$, $\bar{u}_i = -\bar{v}_0$ if $x_i = 0$. Note that the assignments to $\bar{u}_i$ and $\bar{u}_{-i}$ are consistent: one of them is equal to $\bar{v}_0$ and the other to $-\bar{v}_0$.

Let us informally discuss what properties a rounding procedure for Max 2-SAT should satisfy. First, note that if $\bar{u}_i$ is close to $\bar{v}_0$, then the SDP contribution of every constraint of the form $x_i \vee x_j$ (for every j) is close to 1 (in particular, if $\bar{u}_i = \bar{v}_0$, then the contribution is 1). Hence, we want to round $\bar{u}_i$ to 1 with high probability. Similarly, if $\bar{u}_i$ is close to $-\bar{v}_0$, then the SDP contribution of every constraint of the form $x_{-i} \vee x_j$ is close to 1; hence, we want to round $\bar{u}_i$ to 0 with high probability. We get the following heuristic:

Heuristic Rule 1: If $|\langle \bar{u}_i, \bar{v}_0 \rangle|$ is "large", use threshold rounding. Namely, round $\bar{u}_i$ to 1 if $\langle \bar{u}_i, \bar{v}_0 \rangle > 0$; round $\bar{u}_i$ to 0 if $\langle \bar{u}_i, \bar{v}_0 \rangle < 0$.

On the other hand, if $|\langle \bar{u}_i, \bar{v}_0 \rangle|$ is 0 or small (then $\bar{u}_i$ is far from both $\bar{v}_0$ and $-\bar{v}_0$), we get no or little information from $\langle \bar{u}_i, \bar{v}_0 \rangle$ whether to round $\bar{u}_i$ to 1 or 0. Note that the set of vectors $S = \{\bar{u} : \langle \bar{u}, \bar{v}_0 \rangle = 0\}$ is a sphere, which has no distinguished direction. Hence, it is natural to use the Goemans–Williamson rounding procedure for vectors in S.

Heuristic Rule 2: If $\langle \bar{u}_i, \bar{v}_0 \rangle$ is "small', use the Goemans–Williamson algorithm.

Now we need to combine these two heuristics and get a rounding procedure that works for all vectors, including vectors $\bar{u}_i$ for which $|\langle \bar{u}_i, \bar{v}_0 \rangle|$ is neither "small" not "large". Lewin, Livnat, and Zwick [39] use a clever combination of an "outward rotation" and "skewed" rounding to achieve that in their 0.94016-approximation algorithm for Max 2-SAT. We describe a somewhat simpler algorithm by Charikar et al. [15] that satisfies a $(1 - O(\sqrt{\varepsilon}))$ fraction of the constraint given a $(1 - \varepsilon)$ satisfiable instance. The algorithm solves the SDP relaxation and finds vectors $\bar{u}_i$. Let $\varepsilon' = 1 - \mathsf{SDP}/m$ (note that $\varepsilon' \leq \varepsilon$ since $\mathsf{SDP} \geq \mathsf{OPT}$). The algorithm chooses a random Gaussian vector g with independent components distributed as $\mathcal{N}(0, 1)$. For every i, it lets

$$x_i = \begin{cases} 1, & \text{if } \langle \bar{u}_i, \bar{v}_0 + \sqrt{\varepsilon'}\, g \rangle > 0, \\ 0, & \text{if } \langle \bar{u}_i, \bar{v}_0 + \sqrt{\varepsilon'}\, g \rangle < 0. \end{cases}$$

Note that the algorithm always assigns opposite values to x_i and x_{-i}, since $\langle \bar{u}_i, \bar{v}_0 + \sqrt{\varepsilon'}\, g \rangle = -\langle \bar{u}_{-i}, \bar{v}_0 + \sqrt{\varepsilon'}\, g \rangle$. If $\bar{u}_i$ is close to $\bar{v}_0$ or $-\bar{v}_0$, then $\langle \bar{u}_i, \bar{v}_0 \rangle$ is larger in absolute value than $\langle \bar{u}_i, \sqrt{\varepsilon'}\, g \rangle$ with high probability; thus, we essentially use threshold rounding (Heuristic Rule 1); however, if $\langle \bar{u}_i, \bar{v}_0 \rangle = 0$, then the algorithm rounds u_i depending on the sign of $\langle \bar{u}_i, g \rangle$; that is, it uses the Goemans–Williamson rounding (namely, all vectors in the half-space $\{u : \langle u, g \rangle > 0\}$ are rounded to 1; vectors in the half-space $\{u : \langle u, g \rangle < 0\}$ are rounded to 0).

To analyze this algorithm, Charikar et al. upper bound the probability that each constraint $x_i \vee x_j$ is not satisfied. Let the SDP contribution of the constraint $x_i \vee x_j$ be $1 - \varepsilon'_{ij}$; then, the average of all ε'_{ij} is ε'. It is proved in [15] that the probability that $x_i \vee x_j$ is violated is $O(\sqrt{\varepsilon'} + \varepsilon'_{ij}/\sqrt{\varepsilon} + \sqrt{\varepsilon'})$. Averaging over all constraints and using Jensen's inequality, we get that the expected fraction of violated constraints is $O(\sqrt{\varepsilon'}) = O(\sqrt{\varepsilon})$, as required.

Interestingly, this algorithm differs from many other algorithms in that it may violate a constraint $x_i \vee x_j$ even if the SDP contribution of the constraint is equal to 1 (then, $\varepsilon'_{ij} = 0$); loosely speaking, the algorithm violates the constraint even if the SDP "thinks" that the constraint is "certainly" satisfied. In contrast, the Goemans–Williamson algorithm always satisfies a constraint $x_i \neq x_j$ if its contribution $\frac{1}{4}\|\bar{u}_i - \bar{u}_j\|^2$ is 1 (then, vectors $\bar{u}_i$ and $\bar{u}_j$ are antipodal). It turns out that this difference is not accidental: if an algorithm never violates constraints whose SDP contribution is 1, then it can solve instances with hard constraints; however, Guruswami and Lee showed that no polynomial-time algorithm for Max 2-SAT with hard constraints can distinguish between $(1-\varepsilon)$-satisfiable and at-most-ε-satisfiable instances (if UGC is true) [24].

3 Unique Games

In this section, we define the Unique Games problem, overview known results, and describe an algorithm for Unique Games.

▶ **Definition 3** (Unique Games)**.** Unique Games is a constraint satisfaction problem of arity 2, in which every constraint has the form $x_j = \pi_{ij}(x_i)$ for some permutation π_{ij} of the domain D.

Observe that for every fixed value of the variable x_i, there is a *unique* value for the variable x_j that satisfies the constraint between x_i and x_j. Hence, it is easy to find an exact solution for a completely satisfiable instance of Unique Games: We simply guess the value of one variable and then prorogate the values to all other variables (we do this for each connected component of the constraint graphs). However, this algorithm fails even if 1% of all the constraints are violated in the optimal solution. Khot [32] conjectured that if the optimal solution satisfies a $(1-\varepsilon)$ fraction of the constraints, then it is NP-hard to find a solution satisfying even a δ fraction of the constraints. The conjecture is known as Khot's Unique Games Conjecture. We state it formally below.

▶ **Definition 4** (Unique Games Conjecture (UGC) [32])**.** For every positive ε and δ, there exists a d such that given an instance of Unique Games on a domain of size d, it is NP-hard to distinguish between the following two cases:

- There exists a solution satisfying a $(1-\varepsilon)$ fraction of all the constraints.
- Every assignment satisfies at most a δ fraction of all the constraints.

It is unknown whether the conjecture is true or false. However, UGC has proved to be very useful in obtaining hardness of approximation results. Researchers showed very strong hardness of approximation results that rely on UGC for such problems as Vertex Cover [34], Max Cut and Max 2-Lin(q) [33], ordering CSPs [23], and general MAX CSPs [47] (we discuss the last result in Section 6). Today, we do not know how to obtain similar results under weaker complexity assumptions. That is why UGC has gained a lot of popularity in approximation algorithms and hardness of approximation communities. By now, the question whether the conjecture is true of false is one of the major open questions in theoretical computer science.

There are several general purpose approximation algorithms for the problem [32, 51, 22, 14, 16]. The best approximation algorithms by Charikar, Makarychev, and Makarychev [14] and by Chlamtac, Makarychev, and Makarychev [16] find solutions satisfying a $1-O(\sqrt{\varepsilon \log d})$ fraction and $1-O(\varepsilon\sqrt{\log n \log d})$) fraction of all the constraints (respectively) given a $(1-\varepsilon)$-satisfiable instance with $d = |D|$. Khot, Kindler, Mossel, and O'Donnell [33] showed that there is no polynomial-time algorithm that satisfies more than a $1 - c\sqrt{\varepsilon \log d}$ fraction of all the constraints if UGC is true. Thus, the algorithm [14] cannot be improved for general instances of Unique Games if UGC is true. However, there are known better algorithms for special families of Unique Games. Arora et al. [5] showed that UGC does not hold for instances of Unique Games whose constraint graphs are expanders (see also [42]). Kolla, Makarychev, and Makarychev [36] showed that the Unique Games Conjecture does not hold for random and semi-random instances of Unique Games. Finally, Arora, Barak, and Steurer [3] designed a sub-exponential (super-polynomial) algorithm for arbitrary instances of Unique Games. Given a $(1-\varepsilon)$-satisfiable instance, their algorithm finds a solution satisfying a $(1-\varepsilon^c)$ fraction of all the constraints (for some fixed $c > 0$) in time $\exp(dn^{\varepsilon^c})$.

We now present an SDP-based approximation algorithm for Unique Games by Charikar, Makarychev, and Makarychev [14]. The exposition follows the paper by Chlamtac, Makarychev, and Makarychev [16] (see also [9, 40]).

▶ **Theorem 5** (Charikar, Makarychev, Makarychev [14]). *There exists an approximation algorithm that given a $(1-\varepsilon)$-satisfiable instance of Unique Games, finds a solution satisfying a $1-O(\sqrt{\varepsilon \log d})$ fraction of all the constraints.*

The approximation of Theorem 5 cannot be improved if the Unique Games conjecture is true [33]. We prove Theorem 5 using the technique of orthogonal separators [16].

In the next section, we present a standard SDP relaxation for Unique Games (without ℓ_2^2 triangle inequalities). Then, in Section 3.2, we introduce a technique of orthogonal separators. However, we postpone the proof of existence of orthogonal separators to Section 3.4. In Section 3.3, we present the approximation algorithm and prove Theorem 5. Finally, in Section A, we give some useful bounds on the Gaussian distribution.

3.1 SDP Relaxation

We use a standard SDP relaxation for 2CSPs over non-Boolean domain D. We let $G = (V, E)$ be the constraint graph: The vertices of the graph correspond to the variables x_i, the edges correspond to the constraints $x_j = \pi_{ij}(x_i)$. Formally, $V = \{1, \dots, n\}$, $E = \{(i,j) :$ there is a constraint $x_j = \pi_{ij}(x_i)\}$. To simplify the notation, we assume that the graph does not have parallel edges; i.e. there is at most one constraint between every pair of variables x_i and x_j. We note that this restriction on instances can be removed. In the SDP relaxation, we have a vector $\bar{u}_{ia}$ for every vertex $i \in V$ and label $a \in D$. In the *intended integral* solution, the vector $\bar{u}_{ia}$ is the indicator of the event *"$x_i = a$"*. That is, if x_i^* is the optimal solution, then the corresponding integral solution is as follows:

$$\bar{u}^*_{ia} = \begin{cases} 1, & \text{if } x^*_i = a; \\ 0, & \text{otherwise.} \end{cases}$$

Observe that if a constraint (i,j) is satisfied then $\bar{u}^*_{j\pi_{ij}(a)} = \bar{u}^*_{ia}$ for all $a \in D$. If the constraint is violated, then $\bar{u}^*_{ia} = \bar{u}^*_{j\pi_{ij}(a)} = 0$ for all but exactly two a's: $\bar{u}^*_{ix_i} = 1$, but $\bar{u}^*_{j\pi_{ij}(x_i)} = 0$;

and $\bar{u}^*_{jx_j} = 1$, but $\bar{u}^*_{i\pi_{ij}^{-1}(x_j)} = 0$. Thus,

$$\frac{1}{2}\sum_{a\in D}\|\bar{u}^*_{ia} - \bar{u}^*_{j\pi_{ij}(a)}\|^2 = \begin{cases} 0, & \text{if assignment } x^* \text{ satisfies constraint } (i,j);\\ 1, & \text{if assignment } x^* \text{ violates constraint } (i,j).\end{cases}$$

Therefore, the number of violated constraints equals

$$\frac{1}{2}\sum_{(i,j)\in E}\sum_{a\in D}\|\bar{u}^*_{ia} - \bar{u}^*_{j\pi_{ij}(a)}\|^2. \tag{2}$$

Our goal is to minimize this expression. Note that for a fixed variable x_i, one and only one $\bar{u}^*_{ia}$ equals 1. Hence, (1) $\langle \bar{u}^*_{ia}, \bar{u}^*_{ib}\rangle = 0$, if $a \neq b$; and (2) $\sum_{a\in D}\|\bar{u}^*_{ia}\|^2 = 1$. We now write the SDP relaxation (in this SDP, unlike many SDPs we consider in this survey, the objective measures the number of unsatisfied constraints).

$$\begin{aligned}
\text{minimize}\quad & \frac{1}{2}\sum_{(u,v)\in E}\sum_{a\in D}\|\bar{u}_{ia} - \bar{u}_{j\pi_{uv}(a)}\|^2 \\
\text{subject to}\quad & \\
& \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle = 0 && \text{for all } i\in V \text{ and } a\neq b\\
& \sum_{a\in D}\|\bar{u}_{ia}\|^2 = 1 && \text{for all } i\in V
\end{aligned}$$

This is a relaxation, since for $\bar{u}_{ia} = \bar{u}^*_{ia}$, the SDP value equals the number of violated constraints (see (2)); and $\bar{u}^*_{ia}$ is a feasible solution for the SDP.

3.2 Orthogonal Separators – Overview

As we discussed in the introduction, we want to pick one vector $\bar{u}_{ia}$ among vectors $\bar{u}_{i1},\dots,\bar{u}_{id}$ for every variable x_i and set $x_i = a$. The algorithm picks vectors $\bar{u}_{ia}$ in a loop. At every iteration, the algorithm generates a random subset S of all vectors $\{\bar{u}_{ia}\}_{i\in V; a\in D}$. If for a yet not-assigned variable x_i, the set S contains a unique vector $\bar{u}_{ia}$, then the algorithm assigns the value a to x_i. The main technical tool of the algorithm is the procedure for sampling the set S, which we call an orthogonal separator. The distribution of the set S must satisfy certain properties which we describe in this section. One of the main properties of S is as follows (see below for a formal statement): The probability that $\bar{u}_{ia}\in S$ given that $\bar{u}_{ib}\in S$ is small (for $a\neq b$). This property ensures that we rarely pick two different values a and b for a variable x_i. It also explains the name "orthogonal separators": If a vector $\bar{u}_{ia}\in S$, then it is not likely that an *orthogonal vector* $\bar{u}_{ib}$ belongs to S. Thus, the vectors $\bar{u}_{ia}$ and $\bar{u}_{ib}$ are separated by S.

Orthogonal separators are used not only in algorithms for Unique Games, but also in various graph partitioning algorithms [4, 9, 37, 40, 41, 44].

Let X be a set of vectors in ℓ_2 of length at most 1. We say that a random subset $S\subset X$ is an L-orthogonal separator of X with ℓ_2 distortion $\mathcal{D}$, probability scale $\alpha > 0$ and separation threshold $\beta < 1$, if the following conditions hold:

1. For all $\bar{u}\in X$, $\Pr(\bar{u}\in S) = \alpha\|\bar{u}\|^2$.

2. For all $\bar{u},\bar{v}\in X$ with $\langle\bar{u},\bar{v}\rangle \le \beta\max(\|\bar{u}\|^2,\|\bar{v}\|^2)$,

$$\Pr(\bar{u}\in S \text{ and } \bar{v}\in S) \le \frac{\alpha\min(\|\bar{u}\|^2,\|\bar{v}\|^2)}{L}.$$

3. For all $\bar{u}, \bar{v} \in X$,

$$\Pr(I_S(\bar{u}) \neq I_S(\bar{v})) \leq \alpha\mathcal{D}\,\|\bar{u}-\bar{v}\| \cdot \min(\|\bar{u}\|, \|\bar{v}\|) + \alpha\big|\|\bar{u}\|^2 - \|\bar{v}\|^2\big|,$$

where I_S is the indicator of the set S; i.e. $I_S(\bar{u}) = 1$, if $\bar{u} \in S$; $I_S(\bar{u}) = 0$, if $\bar{u} \notin S$.

In most cases, it is convenient to use a slightly weaker (but simpler) bound on $\Pr(I_S(\bar{u}) \neq I_S(\bar{v}))$.

3′. For all $\bar{u}, \bar{v} \in X$,

$$\Pr(I_S(\bar{u}) \neq I_S(\bar{v})) \leq \alpha\mathcal{D}'\|\bar{u}-\bar{v}\| \cdot \max(\|\bar{u}\|, \|\bar{v}\|).$$

The property (3′) follows from (3) with $\mathcal{D}' = \mathcal{D} + 2$, since

$$\big|\|\bar{u}\|^2 - \|\bar{v}\|^2\big| = \big|\|\bar{u}\| - \|\bar{v}\|\big| \cdot (\|\bar{u}\| + \|\bar{v}\|) \leq \|\bar{u}-\bar{v}\| \cdot 2\max(\|\bar{u}\|, \|\bar{v}\|).$$

The last inequality follows from the (regular) triangle inequality for vectors $\bar{u}$, $\bar{v}$ and $(\bar{u}-\bar{v})$.

Our algorithm for Unique Games relies on the following theorem.

▶ **Theorem 6** (see Chlamtac, Makarychev, Makarychev [16]). *There exists a polynomial-time randomized algorithm that given a set of vectors X in the unit ball and parameter L, generates an L-orthogonal separator with ℓ_2 distortion $\mathcal{D} = O\left(\sqrt{\log L}\right)$, probability scale $\alpha \geq poly(1/m)$ and separation threshold $\beta = 0$.*

▶ Remark. Chlamtac, Makarychev, Makarychev [16] proved that there exists an ℓ_2^2 orthogonal separator satisfying conditions (1), (2), and (3″):

3″. For all $\bar{u}, \bar{v} \in X$, $\Pr(I_S(\bar{u}) \neq I_S(\bar{v})) \leq \alpha\tilde{\mathcal{D}}\|\bar{u}-\bar{v}\|^2$, where $\tilde{\mathcal{D}} = O(\sqrt{\log n \log L})$.

If we use this type of orthogonal separators in the algorithm that we present in the next section, we will get an approximation algorithm that satisfies a $1 - O(\varepsilon\sqrt{\log n \log d})$ fraction of the constraints given a $1-\varepsilon$ satisfiable instance.

3.3 Approximation Algorithm

We now present an approximation algorithm for Unique Games that uses orthogonal separators. We prove Theorem 6 and show how to generate orthogonal separators in Section 3.4. Consider the algorithm presented in Figure 4.

▶ **Lemma 7.** *The algorithm satisfies the constraint between variables i and j with probability $1 - O(\mathcal{D}\sqrt{\varepsilon_{ij}})$, where $\mathcal{D}$ is the distortion of the orthogonal separator sampled by the algorithm, and ε_{ij} is the SDP contribution of the term corresponding to the edge (i,j):*

$$\varepsilon_{ij} = \frac{1}{2}\sum_{a \in D} \|\bar{u}_{ia} - \bar{u}_{j\pi_{ij}(a)}\|^2.$$

Proof. If $(\mathcal{D}+2)\sqrt{\varepsilon_{ij}} \geq 1/8$, then the statement holds trivially, so we assume that $(\mathcal{D}+2)\sqrt{\varepsilon_{ij}} < 1/8$. For the sake of analysis, we also assume that π_{ij} is the identity permutation (we can simply rename the values of the variable x_j so that π_{ij} is the identity; this clearly does not affect the execution of the algorithm).

Consider the first iteration in which we assign a value to one of the variables, x_i or x_j. At the end of this iteration, we mark the constraint $x_i = \pi_{ij}(x_j)$ as satisfied or not: If we assign the same value $a \in D$ to x_i and x_j, we mark the constraint as satisfied (recall that we assume that π_{ij} is the identity permutation); otherwise, we conservatively mark the constraint as violated (a constraint marked as violated in the analysis may potentially be satisfied by the algorithm). Consider one iteration of the algorithm at which both x_i and x_j are active. There are three possible cases:

Input: An instance of Unique Games.
Output: Assignment of labels to vertices.

1. Solve the SDP. Let $X = \{\bar{u}_{ia} : i \in V, a \in D\}$.
2. Mark all variables as active.
3. while (there are active variables)
   - **a.** Produce an L-orthogonal separator $S \subset X$ with distortion $\mathcal{D}$ and probability scale α as in Theorem 6, where $L = 4d$ and $\mathcal{D} = O(\sqrt{\log L})$.
   - **b.** For all active variables x_i:
     - Let $S_i = \{a : \bar{u}_{ia} \in S\}$.
     - If S_i contains exactly one element a, then let $x_i = a$; mark the variable x_i as inactive.
4. If the algorithm performs more than n/α iterations, assign arbitrary values to any remaining variables (note that $\alpha \geq 1/poly(d)$).

Figure 4 Approximation algorithm for Unique Games

1. Both sets S_i and S_j are equal and contain exactly one element, then the constraint $x_j = \pi_{ij}(x_i)$ is satisfied after this iteration.
2. The sets S_i and S_j are equal, but contain more than one or none elements, then no values are assigned at this iteration to x_i and x_j.
3. The sets S_i and S_j are not equal, then the constraint may be violated.

Let us estimate the probabilities of each of these events. For a fixed a, we have

$$\begin{aligned}\Pr\left(|S_i| = 1;\ a \in S_i\right) &= \Pr\left(a \in S_u\right) - \Pr\left(a \in S_u \text{ and } b \in S_u \text{ for some } b \neq a\right)\\ &\geq \Pr\left(a \in S_u\right) - \sum_{b \in D \setminus \{a\}} \Pr\left(a, b \in S_u\right).\end{aligned}$$

Using that for all $a \neq b$ the vectors $\bar{u}_{ia}$ and $\bar{u}_{ib}$ are orthogonal, and properties 1 and 2 of orthogonal separators we get (below α is the probability scale):

$$\begin{aligned}\Pr\left(|S_i| = 1;\ a \in S_i\right) &\geq \alpha\|\bar{u}_{ia}\|^2 - \sum_{b \in D \setminus \{a\}} \frac{\alpha \min(\|\bar{u}_{ia}\|^2, \|\bar{u}_{ib}\|^2)}{4d}\\ &\geq \alpha\|\bar{u}_{ia}\|^2 - \frac{\alpha}{4d} \sum_{b \in D \setminus \{a\}} \|\bar{u}_{ia}\|^2\\ &\geq \alpha\|\bar{u}_{ia}\|^2 \Big(1 - \frac{(d-1)}{4d}\Big) \geq \frac{3\alpha\|u_{ia}\|^2}{4}.\end{aligned}$$

Then, using that $\sum_{a \in D} \|\bar{u}_{ia}\|^2 = 1$, we get

$$\Pr\left(|S_i| = 1\right) = \sum_{a \in D} \Pr\left(|S_i| = 1;\ a \in S_i\right) \geq \sum_{a \in D} \frac{3\alpha\|u_{ia}\|^2}{4} = \frac{3\alpha}{4}. \tag{3}$$

Thus, at every iteration of the algorithm when x_i is active, we assign a value to x_i with probability at least $3\alpha/4$. The probability that the constraint (i, j) is violated is at most

$$\Pr\left(S_i \neq S_j\right) \leq \sum_{a \in D} \Pr\left(I_S(\bar{u}_{ia}) \neq I_S(\bar{u}_{ja})\right).$$

We use property 3 of orthogonal separators (see property (3′)) to upper bound the right hand side

$$\Pr\left(S_i \neq S_j\right) \leq \alpha \mathcal{D}' \sum_{a \in D} \|\bar{u}_{ia} - \bar{u}_{ja}\| \cdot \max(\|\bar{u}_{ia}\|, \|\bar{u}_{ja}\|).$$

By Cauchy–Schwarz,

$$\begin{aligned}
\Pr\left(S_u \neq S_v\right) &\leq \alpha \mathcal{D}' \sqrt{\sum_{a \in D} \|\bar{u}_{ia} - \bar{u}_{ja}\|^2} \cdot \sqrt{\sum_{a \in D} \max(\|\bar{u}_{ia}\|^2, \|\bar{u}_{ja}\|^2)} \\
&\leq \alpha \mathcal{D}' \sqrt{\sum_{i \in D} 2\varepsilon_{ij}} \cdot \underbrace{\sqrt{\sum_{a \in D} \|\bar{u}_{ia}\|^2 + \|\bar{u}_{ja}\|^2}}_{=\sqrt{2}} \\
&= 2\alpha \mathcal{D}' \sqrt{\varepsilon_{ij}}.
\end{aligned}$$

Finally, the probability of satisfying the constraint is at least

$$\Pr\left(|S_u| = 1 \text{ and } S_u = S_v\right) \geq \frac{3}{4}\alpha - 2\alpha \mathcal{D}' \sqrt{\varepsilon_{ij}} \geq \frac{1}{2}\,\alpha.$$

Here, we used the assumption $\mathcal{D}'\sqrt{\varepsilon_{ij}} \leq 1/8$. Since the algorithm performs n/α iterations, the probability that it does not assign any value to x_i or x_j before step 4 is exponentially small. At each iteration the probability of failure is at most $O(\mathcal{D}\sqrt{\varepsilon_{ij}})$ times the probability of success, thus the probability that the constraint is not satisfied is $O(\mathcal{D}\sqrt{\varepsilon_{ij}})$. ◀

We now show that the approximation algorithm satisfies a $1 - O(\sqrt{\varepsilon \log d})$ fraction of all the constraints.

Proof of Theorem 5. By Lemma 7, the expected number of unsatisfied constraints is equal to

$$\sum_{(i,j) \in E} O(\sqrt{\varepsilon_{ij} \log d}).$$

By Jensen's inequality for the function $t \mapsto \sqrt{t}$,

$$\frac{1}{|E|} \sum_{(i,j) \in E} \sqrt{\varepsilon_{ij} \log d} \leq \sqrt{\frac{1}{|E|} \sum_{(i,j) \in E} \varepsilon_{ij} \log d} = \sqrt{\frac{\mathsf{SDP}}{|E|} \log d}.$$

Here, $\mathsf{SDP} = \sum_{(i,j) \in E} \varepsilon_{ij}$ denotes the SDP value. If $\mathsf{OPT} \leq \varepsilon|E|$, then $\mathsf{SDP} \leq \mathsf{OPT} \leq \varepsilon|E|$. Hence, the expected cost of solution is upper bounded by $O(\sqrt{\varepsilon \log d})|E|$. ◀

3.4 Orthogonal Separators – Proofs

Proof of Theorem 6. In the proof, we denote the probability that a Gaussian $\mathcal{N}(0,1)$ random variable X is greater than a threshold t by $\bar{\Phi}(t)$. We use the following algorithm for generating L-orthogonal separators with ℓ_2 distortion: Assume w.l.o.g. that all vectors $\bar{u}$ lie in $\mathbb{R}^n$. Fix a threshold $t = \bar{\Phi}^{-1}(1/L)$ (i.e., fix t such that $\bar{\Phi}(t) = 1/L$). Sample independently a random Gaussian n dimensional vector $g \sim \mathcal{N}(0, I)$ in $\mathbb{R}^n$ and a random number r uniformly distributed in $[0, 1]$. Return the set

$$S = \{\bar{u} : \langle \bar{u}, g \rangle \geq t\|\bar{u}\| \text{ and } \|\bar{u}\|^2 \geq r\}.$$

We note that the idea of using threshold rounding was first used by Karger, Motwani, and Sudan [29] in their algorithm for approximate graph coloring. We claim that S is an L-orthogonal separator with ℓ_2 distortion $O(\sqrt{\log L})$, probability scale $\alpha = 1/L$ and $\beta = 0$. Let us verify that S satisfies the required conditions.

1. For every nonzero vector $\bar{u} \in X$, we have

$$\Pr(\bar{u} \in S) = \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\| \text{ and } r \leq \|\bar{u}\|^2) = \\ = \underbrace{\Pr(\langle \bar{u}/\|\bar{u}\|, g\rangle \geq t)}_{1/L} \cdot \underbrace{\Pr(r \leq \|\bar{u}\|^2)}_{\|\bar{u}\|^2} = \|\bar{u}\|^2/L \equiv \alpha\|\bar{u}\|^2.$$

Here we used that $\langle \bar{u}/\|\bar{u}\|, g\rangle$ is distributed as $\mathcal{N}(0,1)$, since $\bar{u}/\|\bar{u}\|$ is a unit vector. Then, by the choice of the threshold t, we have $\Pr(\langle \bar{u}/\|\bar{u}\|, g\rangle \geq t) = 1/L$. If $\bar{u} = 0$, then $\Pr(r \leq \|\bar{u}\|^2) = 0$, hence $\Pr(\bar{u} \in S) = 0$.

2. For every $\bar{u}, \bar{v} \in X$ with $\langle \bar{u}, \bar{v}\rangle = 0$, we have

$$\begin{aligned}\Pr(\bar{u}, \bar{v} \in S) &= \Pr(\langle \bar{u}, g\rangle \geq t;\ \langle \bar{v}, g\rangle \geq t;\ r \leq \|\bar{u}\|^2 \text{ and } r \leq \|\bar{v}\|^2) \\ &= \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\| \text{ and } \langle \bar{v}, g\rangle \geq t\|\bar{v}\|) \cdot \Pr(r \leq \min(\|\bar{u}\|^2, \|\bar{v}\|^2)).\end{aligned}$$

The random variables $\langle \bar{u}, g\rangle$ and $\langle \bar{v}, g\rangle$ are independent, since $\bar{u}$ and $\bar{v}$ are orthogonal vectors. Hence,

$$\Pr(\bar{u}, \bar{v} \in S) = \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|) \cdot \Pr(\langle \bar{v}, g\rangle \geq t\|\bar{v}\|) \cdot \Pr(r \leq \min(\|\bar{u}\|^2, \|\bar{v}\|^2)).$$

Note that $\bar{u}/\|\bar{u}\|$ is a unit vector, and $\langle \bar{u}/\|\bar{u}\|, g\rangle \sim \mathcal{N}(0,1)$. Thus,

$$\Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|) = \Pr(\langle \bar{u}/\|\bar{u}\|, g\rangle \geq t) = 1/L.$$

Similarly, $\Pr(\langle \bar{v}, g\rangle \geq t\|\bar{v}\|) = 1/L$. Then, $\Pr(r \leq \min(\|\bar{u}\|^2, \|\bar{v}\|^2)) = \min(\|\bar{u}\|^2, \|\bar{v}\|^2)$, since r is uniformly distributed in $[0,1]$. We get (note: $\alpha = 1/L$)

$$\Pr(\bar{u}, \bar{v} \in S) = \frac{\min(\|\bar{u}\|^2, \|\bar{v}\|^2)}{L^2} = \frac{\alpha \min(\|\bar{u}\|^2, \|\bar{v}\|^2)}{L}.$$

3. If $I_S(\bar{u}) \neq I_S(\bar{v})$, then either $\bar{u} \in S$ and $\bar{v} \notin S$, or $\bar{u} \notin S$ and $\bar{v} \in S$. Thus,

$$\Pr(I_S(\bar{u}) \neq I_S(\bar{v})) = \Pr(\bar{u} \in S; \bar{v} \notin S) + \Pr(\bar{u} \notin S; \bar{v} \in S).$$

We upper bound the both terms on the right hand side using the following lemma (switching $\bar{u}$ and $\bar{v}$ for the second term) and obtain the desired inequality.

▶ **Lemma 8.** *We have*

$$\Pr(\bar{u} \in S; \bar{v} \notin S) \leq \alpha\mathcal{D}\|\bar{u} - \bar{v}\| \cdot \min(\|\bar{u}\|, \|\bar{v}\|) + \alpha(\|\bar{u}\|^2 - \|\bar{v}\|^2)^+,$$

where $(\cdot)^+$ *denotes the function* $(x)^+ = \max(x, 0)$.

Proof of Lemma 8. We have

$$\Pr(\bar{u} \in S; \bar{v} \notin S) = \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|;\ r \leq \|\bar{u}\|^2;\ v \notin S).$$

The event $\{\bar{v} \notin S\}$ is the union of two events $\{\langle \bar{v}, g\rangle \geq t\|\bar{v}\| \text{ and } r \leq \|\bar{v}\|^2\}$ and $\{r \geq \|\bar{v}\|^2\}$. Hence,

$$\begin{aligned}\Pr(\bar{u} \in S; \bar{v} \notin S) \leq\ & \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|;\ \langle \bar{v}, g\rangle < t\|\bar{v}\|;\ r \leq \min(\|\bar{u}\|^2, \|\bar{u}\|^2)) \\ &+ \Pr(\langle \bar{u}, g\rangle \geq t\|\bar{u}\|;\ \|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2).\end{aligned} \tag{4}$$

Let $g_u = \langle \bar{u}/\|\bar{u}\|, g\rangle$ and $g_v = \langle \bar{v}/\|\bar{v}\|, g\rangle$. Both g_u and g_v are standard $\mathcal{N}(0,1)$ Gaussian random variables. Thus, $\Pr(g_u \geq t) = \Pr(g_v \geq t) = 1/L = \alpha$. We write (4) as follows:

$$\begin{aligned}\Pr(\bar{u} \in S; \bar{v} \notin S) &= \Pr(g_u \geq t;\ g_v < t)\Pr(r \leq \min(\|\bar{u}\|^2, \|\bar{v}\|^2)) + \Pr(g_u \geq t)\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2)\\ &= \Pr(g_u \geq t;\ g_v < t)\cdot \min(\|\bar{u}\|^2, \|\bar{v}\|^2) + \alpha \Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2). \qquad (5)\end{aligned}$$

To finish the proof we need to estimate $\Pr(g_u \geq t;\ g_v < t)$ and $\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2)$. Since r is uniformly distributed in $[0,1]$, $\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2) = \|\bar{u}\|^2 - \|\bar{v}\|^2$, if $\|\bar{u}\|^2 - \|\bar{v}\|^2 > 0$; and $\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2) = 0$, otherwise. That is, $\Pr(\|\bar{v}\|^2 \leq r \leq \|\bar{u}\|^2) = (\|\bar{u}\|^2 - \|\bar{v}\|^2)^+$.

We use Lemma 27 to upper bound $\Pr(g_u \geq t;\ g_v < t)$:

$$\Pr(g_u \geq t;\ g_v < t) \leq O\big(\sqrt{1-\operatorname{cov}(g_u, g_v)}\cdot\sqrt{\log L}/L\big). \qquad (6)$$

The covariance of g_u and g_v equals $\operatorname{cov}(g_u, g_v) = \langle \bar{u}/\|\bar{u}\|, \bar{v}/\|\bar{v}\|\rangle$ and $\|\bar{u} - \bar{v}\|^2 = \|\bar{u}\|^2 + \|\bar{v}\|^2 - 2\langle \bar{u}, \bar{v}\rangle$. Hence,

$$\begin{aligned}1 - \operatorname{cov}(g_u, g_v) &= 1 - \frac{\|\bar{u}\|^2 + \|\bar{v}\|^2 - \|\bar{u}-\bar{v}\|^2}{2\|\bar{u}\|\,\|\bar{v}\|} = \frac{\|\bar{u}-\bar{v}\|^2 - (\|\bar{u}\|^2 + \|\bar{v}\|^2 - 2\|\bar{u}\|\,\|\bar{v}\|)}{2\|\bar{u}\|\,\|\bar{v}\|}\\ &= \frac{\|\bar{u}-\bar{v}\|^2 - (\|\bar{u}\| - \|\bar{v}\|)^2}{2\|\bar{u}\|\,\|\bar{v}\|} \leq \frac{\|\bar{u}-\bar{v}\|^2}{2\|\bar{u}\|\,\|\bar{v}\|}.\end{aligned}$$

We plug this bound in (6) and get

$$\Pr(g_u \geq t;\ g_v < t) \leq \alpha \cdot \frac{\|\bar{u}-\bar{v}\|}{\sqrt{\|\bar{u}\|\,\|\bar{v}\|}} \cdot O(\sqrt{\log L}) \leq \alpha \cdot \frac{\|\bar{u}-\bar{v}\|}{\min(\|\bar{u}\|, \|\bar{v}\|)} \cdot O(\sqrt{\log L}).$$

Now, Lemma 8 follows from (5). This concludes the proof of Lemma 8 and Theorem 6. ◀

◀

4 CSPs of higher arities

In this section, we discuss techniques for solving Max k-CSP(d). We will not present any approximation algorithms for Max k-CSP(d), but rather we will describe an SDP relaxation for the problem and explain why rounding this SDP is challenging. To be specific, we will focus our attention on the regime when $k > \Omega(\log d)$. In this regime, the best known approximation is $\Omega(dk/d^k)$ [43].

Consider an instance of Max k-CSP(d). As we noted in the introduction, we may assume that all constraints are of the form $(x_{i_1} = j_1) \wedge \cdots \wedge (x_{i_k} = j_k)$. We write an SDP relaxation for the problem. In the SDP, we have two sets of variables. First, we have a variable $\bar{u}_{ij}$ for each x_i and $j \in D$; second, we have a variable $\bar{v}_C = \bar{v}_{(i_1,j_1),\ldots,(i_k,j_k)}$ for each constraint C of the form $(x_{i_1} = j_1) \wedge \cdots \wedge (x_{i_k} = j_k)$. We denote the set of the constraints by $\mathcal{C}$.

$$\text{maximize} \sum_{C\in\mathcal{C}} \|\bar{v}_C\|^2$$

subject to

$$\begin{aligned}
&\sum_{j=1}^{d} \|\bar{u}_{ij}\|^2 = 1 && \text{for every } i && (7)\\
&\langle \bar{u}_{ij_1}, \bar{u}_{ij_2}\rangle = 0 && \text{for every } i \text{ and } j_1 \neq j_2 &&\\
&\langle \bar{u}_{ij}, \bar{v}_C\rangle = \|\bar{v}_C\|^2 && \text{for every } C \in \mathcal{C} \text{ and clause } x_i = j \text{ in } C &&\\
&\langle \bar{u}_{ij}, \bar{v}_C\rangle = 0 && \text{for every } C \in \mathcal{C} \text{ and clause } x_i = j \text{ not in } C && (8)
\end{aligned}$$

In the intended solution, for some unit vector $\bar{v}_0$, we have $\bar{u}_{ij} = \bar{v}_0$ if $x_i = j$, and $\bar{u}_{ij} = 0$, otherwise; $\bar{v}_C = \bar{v}_0$ if C is satisfied, and $\bar{v}_C = 0$, otherwise. Let us first consider a very basic rounding algorithm for the problem and discuss when it works and when it does not:

1. Choose a random Gaussian vector g with independent components distributed as $\mathcal{N}(0,1)$.
2. For every i, let $x_i = \arg\max_j |\langle \bar{u}_{ij}, g\rangle|$.

We analyze the algorithm. Let us consider a CSP constraint $C \in \mathcal{C}$ and estimate with what probability the algorithm satisfies it. Keep in mind that we want to get an $\Omega(kd/d^k)$ approximation, so the desired probability is $\Omega(kd\,|\bar{v}_C|^2/d^k)$. By renaming variables and values, we may assume that C is $(x_1 = 1) \wedge \cdots \wedge (x_k = 1)$. Denote $\xi_{ij} = \langle \bar{u}_{ij}, g\rangle$. Note that ξ_{ij} is a Gaussian random variable with mean 0 and variance $\|\bar{u}_{ij}\|^2$. The probability that C is satisfied equals $\Pr(|\xi_{i1}| > |\xi_{ij}| \text{ for all } i \text{ and } j \neq 1)$. It is instructive to consider a very special case when the following two assumptions hold.

1. Since $\langle \bar{u}_{i1}, \bar{v}_C\rangle = \|\bar{v}_c\|^2$ for all i, we can write $\bar{u}_{i1} = \bar{v}_C + \bar{u}_{i1}^{\perp}$ where $\bar{u}_{i1}^{\perp} \perp \bar{v}_C$. Let us assume that all vectors $\bar{u}_{i1}^{\perp}$ are equal to 0, and thus $\bar{u}_{i1} = \cdots = \bar{u}_{id} = \bar{v}_C$.
2. Let us assume that all vectors $\bar{u}_{ij}$ with $j \neq 1$ have the same length.

The first assumption is not essential, and we make it to slightly simplify the computations; however, the second assumption is crucial for our analysis. We have, $\xi_{11} = \cdots = \xi_{k1}$. Let $\xi = \xi_{i1}$, η be a random variable distributed as $\mathcal{N}(0, \|\bar{u}_{ij}\|^2)$, and $\rho = \operatorname{Var}[\xi] \,/\, \operatorname{Var}[\eta] = \|\bar{v}_C\|^2/\|\bar{u}_{ij}\|^2$, where $j \neq 1$. Assume that $\rho \leq 1$. It follows from (8) that all random variables ξ_{ij}, with $j \neq 1$, are independent from ξ. Thus, for every M,

$$\begin{aligned}
&\Pr(|\xi| > |\xi_{ij}| \text{ for all } i \text{ and } j \neq 1) \geq \Pr(|\xi| \geq M \text{ and } |\xi_{ij}| < M \text{ for all } i \text{ and } j \neq 1)\\
&\quad = \Pr(|\xi| \geq M)\Pr(|\xi_{ij}| < M \text{ for } i \text{ and } j \neq 1) \overset{\text{Šidák}}{\geq} \Pr(|\xi| \geq M) \prod_{i;\, j\neq 1} \Pr(|\xi_{ij}| < M)\\
&\quad = \Pr(|\xi| \geq M)\Pr(|\eta| < M)^{k(d-1)}.
\end{aligned}$$

Here, we used Šidák's theorem to get the inequality on the second line. From Lemma 26, it is easy to prove that $\Pr(|\eta| \geq M) \leq \Pr(|\xi| \geq M)^{\rho}$ for every threshold M (see [43, Lemma 2.4]). Let $p = 1/(\rho k(d-1))$ and M be such that $\Pr(|\eta| \geq M) = p$. Then the probability that the constraint is satisfied is at least

$$p^{1/\rho}(1-p)^{k(d-1)} \approx \left(\frac{1}{\rho k(d-1)}\right)^{1/\rho} e^{-pk(d-1)} \approx \left(\frac{1}{\rho kd}\right)^{1/\rho} e^{-1/\rho} = \left(\frac{1}{e\rho kd}\right)^{1/\rho}. \tag{9}$$

When $\rho > c/k$ (for sufficiently large c), the probability of satisfying C is, loosely speaking, of order $1/d^{k/c}$, which is much greater than the desired probability $dk\|\bar{v}_C\|^2/d^k$. On the other hand, when $\rho < c/k$, we can use the random assignment rounding. Indeed, from (7) and our assumptions, we get

$$1 = \|\bar{u}_{i1}\|^2 + \sum_j \|\bar{u}_{ij}\|^2 = \|\bar{v}_C\|^2 + \frac{d-1}{\rho}\|\bar{v}_C\|^2.$$

Thus, $\|\bar{v}_C\|^2 \approx \rho/d < c/(dk)$ and the desired probability of satisfying the constraint is $O(1/d^k)$, which is less than the probability $1/d^k$ with which the random assignment algorithm satisfies the constraint (when constants in the $O(\cdot)$-notation are appropriately chosen). We see that if choose uniformly at random one of the two algorithms, the basic SDP rounding and the random assignment, we will satisfy every clause with the desired probability.

This argument can be made formal. However, it crucially uses that all vectors $\bar{u}_{ij}$, for $j \neq 1$, have the same length. If some of the vectors, are considerably longer than $1/\sqrt{d}$,

they will be chosen disproportionately often. For instance, assume that $\|v_C\|^2 = c/(dk)$, a δ fraction of vectors $\bar{u}_{ij}$ have length approximately $1/\sqrt{\delta d}$, and the remaining vectors $\bar{u}_{ij}$ are equal to 0. In this setting, there are $d' \approx \delta d$ non-zero vectors $\bar{u}_{ij}$ for every i, and $\rho' = \|\bar{v}_C\|^2/\|\bar{u}_{ij}\|^2 \approx c\delta/k$ (for $j > 1$ such that $\bar{u}_{ij} \neq 0$). Now, loosely speaking, we can use formula (9) with $\rho = \rho'$ and $d = d'$ to estimate the probability of satisfying the constraint[5]. We get that the probability is approximately $1/d^{k/(c\delta)}$, which is much less than the desired probability when $\delta \ll 1/c$.

Let us discuss how we can fix the algorithm. If we knew that $\|\bar{u}_{i1}\|^2 \leq O(1/d)$ for all i, we would be able to restrict our attention only to those values j such that $\|\bar{u}_{ij}\|^2 \leq O(1/d)$; that is, for each i, we would choose j that maximizes $|\xi_{ij}|$ only among those j for which $\|\bar{u}_{ij}\|^2 \leq O(1/d)$. That would eliminate values j that the rounding procedure chooses with disproportionately large probabilities and thus fix the algorithm. On the other hand, if we knew that $\|\bar{u}_{i1}\|^2 > c/d$ for all i (for some sufficiently large constant $c > 1$), then we would be able to find a good assignment using another algorithm: for every i, we would choose j uniformly at random among those j for which $\|\bar{u}_{ij}\|^2 > c/d$. Note that for every i, there are at most d/c such values of j. Therefore, the probability that we choose $j = 1$ for every i is at least $(c/d)^k \gg dk/d^k$, as desired. However, we may have $\|\bar{u}_{i1}\|^2 < c/d$ for some values of i and $\|\bar{u}_{i1}\|^2 > c/d$ for other values of i. Nevertheless, it is shown in [43] that it is possible to combine these two approaches and get an $\Omega(dk/d^k)$ approximation (when $k = \Omega(\log d)$). We refer the reader to [43] for details.

5 Minimum Multiway Cut

In this section, we describe known approximation results for the Minimum Multiway Cut problem. From a CSP viewpoint, the problem is a CSP of arity 2 over a domain D with two types of constraints:

- Equality constraints of the form $x_i = x_j$.
- Unary constraints of the form $x_i = j$, where $j \in D$.

The objective is to minimize the number of unsatisfied constraints. The problem is usually stated as a graph partitioning problem.

▶ **Definition 9.** We are given a graph $G = (V, E)$ and a set of terminals $T = \{s_1, \ldots, s_d\} \subset V$. We need to partition the graph into d pieces $P_1, \ldots, P_d$ such that $s_i \in P_i$. Our goal is to minimize the number of cut edges.

Observe that the two formulations are equivalent. Given a CSP instance, we construct a graph instance of the problem as follows: we introduce a vertex v_i for each variable x_i and add auxiliary vertices $s_1, \ldots, s_d$; for each constraint $x_{i_1} = x_{i_2}$, we add an edge (v_{i_1}, v_{i_2}); for each constraint $x_i = j$ we add an edge (v_i, s_j). Similarly, we can transform a graph instance to a CSP instance. Note that there is a one-to-one correspondence between solutions to the problems: an assignment $A : \{x_i\} \to D$ corresponds to the partitioning $P_1, \ldots, P_d$ with $P_j = \{v_i : A(x_i) = j\}$ and vice versa. Below, we will discuss the Multiway Cut problem in the standard graph partitioning formulation. Note that the problem can be solved in polynomial-time when $d = 2$ (then, it is equivalent to the Minimum Cut problem), but it is NP-hard for every $d \geq 3$ [19].

[5] We showed that formula (9) is a lower bound on the probability that C is satisfied; but, in fact, it gives a reasonably accurate estimate on the probability.

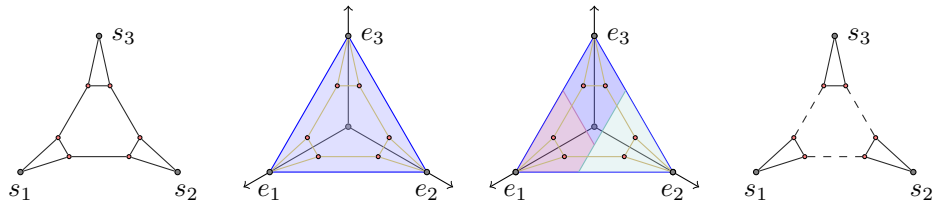


Figure 5 The figure shows (1) an input graph, (2) a feasible LP solution, (3) a random partitioning, and (4) the corresponding cut in the graph.

Minimum Multiway Cut was introduced by Dahlhaus, Johnson, Papadimitriou, Seymour, and Yannakakis [19] in 1994. Since then, a number of approximation algorithms have been proposed in the literature [19, 12, 30, 10, 50]. We describe only the latest results. In 2014, Sharma and Vondrák [50] presented 1.309017 and 1.2965 approximation algorithms; the analysis of the latter algorithm is only computer-verified. Most recently (the paper will appear in 2017), Buchbinder, Schwartz, and Weizman [11] designed and analytically analyzed a simpler 1.29694-approximation algorithm.

Freund and Karloff [20] proved an integrality gap of $8/(7+1/(d-1)) = 8/7 - o(1)$ for the problem (where the term $o(1)$ goes to 0 as $d \to \infty$). Most recently, Angelidakis, Makarychev, and Manurangsi improved the gap to $6/(5+1/(d-1)) - \varepsilon = 6/5 - o(1)$ [2]. Manokaran, Naor, Raghavendra, and Schwartz [45] proved that the hardness of approximation factor for Minimum Multiway Cut equals the integrality gap if UGC holds. Thus, these results [20, 45] imply a $6/5 - o(1)$ hardness of approximation (if UGC holds).

Let us also mention the results by Karger, Klein, Stein, Thorup, and Young [30] and Cunningham and Tang [18] for the problem with $d = 3$ terminals. They presented a 12/11-approximation algorithm and showed a matching integrality gap (the integrality gap implies a $(12/11 - \varepsilon)$-UGC-hardness of approximation, as was shown by Manokaran et al. [45]).

We note that the maximization version of the problem, in which the objective is to maximize the number of satisfied constraints received much less attention. Langberg, Rabani, and Swamy [38] designed a $(2+\sqrt{2})/4 \approx 0.85355$ approximation algorithm for the problem and showed an integrality gap of $6/7 - o(1) \approx 0.85714 - o(1)$.

All known approximation algorithms for Minimum Multiway Cut — other than the 2-approximation algorithm by Dahlhaus et al. — use the linear programming (LP) relaxation by Călinescu, Karloff, and Rabani [12]. The recent algorithm by Sharma and Vondrák [50] is significantly more involved than the algorithm by Călinescu et al. and, in particular, requires computer-assisted fine tuning of the parameters to get the best approximation factor. Thus, in this survey, we describe the original LP relaxation and algorithm by Călinescu, Karloff, and Rabani [12].

Consider the following relaxation for the problem. For every vertex u, we introduce d LP variables $u_1, \dots, u_d$ and let $\bar{u} = (u_1, \dots, u_d)$. Let $\Delta = \{\bar{x} \in \mathbb{R}^d : x_1 + \dots + x_d = 1, x_1 \geq 0, \dots, x_d \geq 0\}$; Δ is a simplex with vertices $e_1 = (1, 0, \dots, 0)$, $e_2 = (0, 1, 0, \dots, 0)$, $\dots$, $e_d = (0, \dots, 0, 1)$ (see Figure 5). We write the following linear program.

$$\text{minimize } \frac{1}{2}\sum_{(u,v)\in E} \|\bar{u}-\bar{v}\|_1$$

subject to

$$\begin{aligned} &\bar{s}_j = e_j && \text{for every } j \in \{1,\dots,d\}, \\ &\bar{u} \in \Delta && \text{for every vertex } u. \end{aligned}$$

It is easy to see that this program is indeed a linear program – we can write the objective function as a linear function by introducing auxiliary variables. Namely, introduce additional LP variables y_{uvi}; add inequalities $y_{uvi} \geq u_i - v_i$ and $y_{uvi} \geq v_i - u_i$. Then, replace each term $\|\bar{u}-\bar{v}\|_1$ in the objective function with the expression $\sum_{i=1}^{k} y_{uvi}$.

We denote the value of an optimal solution by OPT, and the value of LP by LP.

▶ **Claim 10.** *The LP is a relaxation of the problem. That is,* $\mathsf{LP} \leq \mathsf{OPT}$.

Proof. Consider an optimal solution $P_1,\dots,P_d$. Let $\bar{u} = e_i$ for $u \in P_i$. Clearly, $\bar{u}$ is a feasible LP solution. We compute the value of this solution. Consider an edge $e = (u,v)$. Suppose that $u \in P_i$ and $v \in P_j$. The contribution of e to the objective function is

$$\frac{\|u-v\|_1}{2} = \frac{\|e_i - e_j\|_1}{2} = \begin{cases} 1, & \text{if } i \neq j, \\ 0, & \text{if } i = j. \end{cases}$$

That is, the contribution of e is 1 if e is cut, and 0, otherwise. Thus, the total contribution of all edges equals the number of cut edges. We get that the value of the LP solution $\{\bar{u}\}$ is OPT. Therefore, $\mathsf{LP} \leq \mathsf{OPT}$. ◀

▶ **Definition 11.** Given a feasible LP solution, we define a distance function $d(\cdot,\cdot)$ on the vertices of the graph:

$$d(u,v) = \frac{1}{2}\|\bar{u}-\bar{v}\|_1.$$

Let $B_r(u)$ be the ball of radius r around vertex u w. r. t. distance $d(\cdot,\cdot)$: $B_r(u) = \{v : d(u,v) < r\}$.

Now, we describe the algorithm by Călinescu, Karloff, and Rabani [12].

▶ **Theorem 12** (Călinescu, Karloff, and Rabani [12])**.** *There exists a* 3/2*-approximation algorithm for Minimum Multiway Cut.*

Proof. The algorithm is presented in Figure 6. We prove that the algorithm always returns a feasible solution.

▶ **Claim 13.** *The algorithm returns a feasible solution.*

Proof. Note that $s_i \in B_r(s_i)$ and $s_i \notin B_r(s_j)$ for every $j \neq i$. Therefore, s_i necessarily lies in P_i, as required. ◀

Now we compute the expected cost of the solution. Consider an edge $e = (u,v)$. Note that

$$d(u,s_i) = \frac{\|\bar{u}-e_i\|_1}{2} = \frac{(1-u_i)+\sum_{j\neq i} u_j}{2} = \frac{(1-u_i)+(1-u_i)}{2} = 1-u_i.$$

Let

$$A_i = \min(d(u,s_i), d(v,s_i)) \quad \text{and} \quad B_i = \max(d(u,s_i), d(v,s_i)).$$

Approximation Algorithm for Multiway Cut

Input: graph $G = (V, E)$ and a set of terminals $T = \{s_1, \dots, s_d\} \subset V$.
Output: a partition $P_1, \dots, P_d$ of V such that $s_i \in P_i$.

solve the LP relaxation for Minimum Multiway Cut. Define $d(u, v) = \frac{1}{2}\|\bar{u} - \bar{v}\|_1$.
choose a random permutation π of $\{1, \dots, d\}$
choose r uniformly at random from $(0, 1)$
let $A = \varnothing$
for $i = \pi(1), \pi(2), \dots, \pi(d-1)$ do
  let $P_i = B_r(s_i) \setminus A$
  let $A = A \cup P_i$
let $P_{\pi(d)} = V \setminus A$
return partition P_1, …, P_d

Figure 6 Approximation algorithm for Multiway Cut

We have,

$$B_i - A_i = |d(u, s_i) - d(v, s_i)| = |u_i - v_i|.$$

We may assume without loss of generality that

$$A_1 \le A_2 \le \dots \le A_d. \tag{10}$$

Let us write $i \prec j$ if $\pi^{-1}(i) < \pi^{-1}(j)$ (i precedes j in the order defined by π). We say that an index i settles the edge (u, v) if i is the first index w.r.t. π such that $u \in B_r(s_i)$ or $v \in B_r(s_i)$ (or both). In other words, index i settles edge e if $A_i \le r$ and $A_j > r$ for all $j \prec i$. Let $\mathcal{E}_i$ be the event that i settles (u, v). Note that at most one of the events $\mathcal{E}_i$ happens. (If no event $\mathcal{E}_i$ happens than u and v belong to $P_{\pi(k)}$, and the edge (u, v) is not cut.)

When the event $\mathcal{E}_i$ happens, we add either one or both of the vertices u and v to P_i. Note that in the former case, the edge (u, v) is cut since $u \in P_i$ and $v \notin P_i$; in the latter case, the edge (u, v) is not cut since $u, v \in P_i$. We conclude that

$$\Pr(e \text{ is cut}) = \sum_{i=1}^{d-1} \Pr(\mathcal{E}_i \text{ and } |\{u, v\} \cap B_r(s_i)| = 1) = \sum_{i=1}^{d-1} \Pr(\mathcal{E}_i \text{ and } A_i \le r < B_i).$$

We are going to show now that $\Pr(\mathcal{E}_i | r) \le 1/2$ for every $i > 1$. Consider the event $\mathcal{L}_i$ that $i \succ 1$. Since events $i \succ 1$ and $1 \succ i$ are equiprobable, event $\mathcal{L}_i$ happens with probability $1/2$. We claim that when $\mathcal{L}_i$ happens $\mathcal{E}_i$ does not happen, and, therefore,

$$\Pr(\mathcal{E}_i | r) \le 1 - \Pr(\mathcal{L}_i | r) = \frac{1}{2}.$$

Assume to the contrary that both events happen. Then

- $r \ge A_i$ and $r < A_j$ for every $j \prec i$,
- and $1 \prec i$,

therefore, $r \ge A_i$ and $r < A_1$; thus, $A_i < A_1$, which contradicts to our assumption (10). We

have,

$$\begin{aligned}\Pr(e \text{ is cut}) &= \sum_{i=1}^{d-1} \Pr(\mathcal{E}_i \text{ and } A_i \le r < B_i) = \sum_{i=1}^{d-1} \mathbb{E}_r\left[\Pr(\mathcal{E}_i|r)\Pr(A_i \le r < B_i|r)\right] \\ &\le (B_1 - A_1) + \sum_{i=2}^{k} \frac{B_i - A_i}{2} = \frac{B_1 - A_1}{2} + \sum_{i=1}^{k} \frac{B_i - A_i}{2} \\ &= \frac{|u_1 - v_1|}{2} + \sum_{i=1}^{k} \frac{|u_i - v_i|}{2} = \frac{|u_1 - v_1| + \|\bar{u} - \bar{v}\|_1}{2}.\end{aligned}$$

Observe that

$$\|u - v\|_1 \ge |u_1 - v_1| + \left|\sum_{i=2}^{k} u_i - \sum_{i=2}^{k} v_i\right| = |u_1 - v_1| + |(1 - u_1) - (1 - v_1)| = 2\,|u_1 - v_1|.$$

Thus $\Pr(e \text{ is cut}) \le 3\|\bar{u} - \bar{v}\|_1/4$. By the linearity of expectation, the expected number of cut edges is at most

$$\sum_{(u,v)\in E} \frac{3}{4}\|\bar{u} - \bar{v}\|_1 = \frac{3\,\mathsf{LP}}{2} \le \frac{3\,\mathsf{OPT}}{2}.$$

We proved that our algorithm gives a 3/2 approximation, in expectation. By running this algorithm many times we can get a $(3/2 + \varepsilon)$ approximation with high probability. In fact, the algorithm can be easily derandomized using the method of conditional expectations. ◀

6 Universal Rounding Algorithm

In this section, we discuss the hardness of approximation result by Raghavendra [47] and universal rounding algorithm by Raghavendra and Steurer [48]. Then, we describe in detail the rounding algorithm for a special case of CSPs of arity 2.

We consider a class of *generalized* CSPs of arity k with variables over a fixed domain $D = \{1, \dots, d\}$ and predicates from a constant-size set Λ. Every predicate $\pi \in \Lambda$ is a function from D^k to $[-1, 1]$. We shall refer to this class of CSPs as k-CSP Λ. The value of a solution x_i^* for instance $\mathcal{I}$ of k-CSP Λ equals

$$\frac{1}{|\Pi|} \sum_{\pi(x_{i_1},\dots,x_{i_k})\in\Pi} \pi(x_{i_1}^*, \dots, x_{i_k}^*),$$

where Π is the set of predicates in $\mathcal{I}$. The expression above has a normalization factor $1/|\Pi|$; thus the value of any solution lies in the range $[-1, 1]$. Note that a regular (non-generalized) CSP is simply a generalized CSP in which all predicates take only values 0 and 1. We say that a predicate π is nonnegative if π is nonnegative on every assignment. Observe that finding an assignment of maximum value in an instance of k-CSP Λ is equivalent to finding an assignment of minimum value in the instance where every predicate π is replaced with $-\pi$. Thus, all results stated for maximization versions of *generalized* CSPs can be easily translated to results for minimization versions of *generalized* CSPs and vice versa. However, as we discuss later, the results for minimization and maximization CSPs with nonnegative predicates – and, in particular, results for minimization and maximization regular CSPs – are quite different.

In a breakthrough paper [47], Raghavendra showed that assuming the Unique Games Conjecture, the best possible approximation for many CSPs can be obtained by solving a

standard SDP relaxation. To formally state his result, we describe the SDP relaxation and introduce some notation. First, we formulate the SDP for CSPs of arity 2; then, in the next section, we present the SDP for CSPs of higher arities. As we discussed earlier, for each variable x_i, we introduce SDP vector variables $\bar{u}_{i1}, \dots, \bar{u}_{id}$. We also introduce a special unit vector $\bar{v}_0$.

$$\text{maximize } \frac{1}{|\Pi|} \sum_{\pi(x_i,x_j)\in\Pi} \sum_{a,b\in D} \pi(a,b)\langle \bar{u}_{ia}, \bar{u}_{jb}\rangle \tag{11}$$

$$\text{subject to} \tag{12}$$

$$\sum_{a\in D} \bar{u}_{ia} = \bar{v}_0 \qquad \text{for all } i \tag{13}$$

$$\langle \bar{u}_{ia}, \bar{u}_{ib}\rangle = 0 \qquad \text{for all } i,j \in [n]; a \neq b \tag{14}$$

$$\langle \bar{u}_{ia}, \bar{u}_{jb}\rangle \geq 0 \qquad \text{for each constraint } \pi(x_i,x_j) \in \Pi \text{ and } a,b \in D \tag{15}$$

Let $\mathsf{OPT} = \mathsf{opt}(\mathcal{I})$ be the value of the optimal solution for instance $\mathcal{I}$, and $\mathsf{SDP} = \mathsf{sdp}(\mathcal{I})$ be the value of the optimal SDP solution for $\mathcal{I}$. Define gap as follows:

$$\mathsf{gap}(s) = \inf\{\mathsf{opt}(\mathcal{I}) : \mathsf{sdp}(\mathcal{I}) \geq s\}.$$

That is, $\mathsf{gap}(s)$ equals the minimum possible optimum value of an instance $\mathcal{I}$ with SDP value at least s. Note that $\mathsf{opt}(\mathcal{I}) \leq \mathsf{sdp}(\mathcal{I})$ for all $\mathcal{I}$, thus if $\inf\{\mathsf{sdp}(\mathcal{I}) : \mathsf{sdp}(\mathcal{I}) \geq s\} = s$, then $\mathsf{gap}(s) \leq s$.

▶ **Theorem 14** (Raghavendra [47]). *Assuming the Unique Games Conjecture, for every positive ε and every $s \in (-1,1)$, it is NP-hard to distinguish between instances $\mathcal{I}$ with $\mathsf{opt}(\mathcal{I}) \geq s$ and $\mathsf{opt}(\mathcal{I}) \leq \mathsf{gap}(s+\varepsilon)+\varepsilon$.*

Note that since we can find the optimal solution of the SDP in polynomial time, we can distinguish instances with (A) $\mathsf{opt}(\mathcal{I}) \geq s$ and (B) $\mathsf{opt}(\mathcal{I}) < \mathsf{gap}(s) - \varepsilon$: If $\mathsf{sdp}(\mathcal{I}) \geq s$, then $\mathsf{opt}(\mathcal{I}) \geq \mathsf{gap}(s)$ and thus $\mathcal{I}$ is in the set (A); if $\mathsf{sdp}(\mathcal{I}) < s$, then $\mathsf{opt}(\mathcal{I}) \leq \mathsf{sdp}(\mathcal{I}) < s$ and thus $\mathcal{I}$ is in the set (B). Furthermore, Raghavendra [47] and then, Raghavendra and Steurer [48] showed that given an instance with SDP value at least s, we can find an assignment of value at least $\mathsf{gap}(s-\varepsilon)-\varepsilon$. We present a proof of this theorem for 2-CSPs with nonnegative predicates in Section 6.2.

▶ **Theorem 15** (Raghavendra and Steurer [48]). *For every class of problems k-CSP Λ and every positive ε, there exists a randomized polynomial time algorithm that given an instance $\mathcal{I}$ of k-CSP Λ with the SDP value SDP finds a solution $\{x_i^*\}$ of value at least $\mathsf{gap}(\mathsf{SDP}-\varepsilon)-\varepsilon$.*

This result applies to both minimization and maximization problems. For a minimization problem, the algorithm by Raghavendra and Steurer [48] finds a solution of cost at most $\mathsf{gap}_{\mathsf{min}}(\mathsf{SDP}+\varepsilon)+\varepsilon$, where $\mathsf{gap}_{\mathsf{min}}(s) = \sup\{\mathsf{opt}(\mathcal{I}) : \mathsf{sdp}(\mathcal{I}) \leq s\}$.

Let us discuss what the results by Raghavendra [47] and Raghavendra and Steurer [48] imply for the three objectives we introduced in the introduction. Consider a generalized k-CSP Λ, and let α be the integrality gap of its SDP relaxation. Can we get an $(\alpha-\varepsilon)$ approximation using the algorithm by Raghavendra and Steurer [48]? Generally speaking, no. If the value of the optimal solution OPT is positive, but is very close to 0, then $\mathsf{gap}(\mathsf{SDP}-\varepsilon)-\varepsilon$ may be negative. Hence, we cannot even guarantee that the algorithm finds a solution of positive value.

Now, assume that all predicates $\pi \in \Lambda$ are nonnegative (in particular, this condition holds for regular CSPs), then the optimal value of any maximization k-CSP Λ is bounded

away from 0 by some positive β (as we will see in a moment) and thus the algorithm always returns a solution of value at least

$$\alpha(\mathsf{SDP} - \varepsilon) - \varepsilon \geq \alpha\,\mathsf{OPT} - 2\varepsilon = \alpha\,\mathsf{OPT} - 2\beta(\varepsilon/\beta) \geq (\alpha - 2\varepsilon/\beta)\,\mathsf{OPT},$$

which is an $(\alpha - 2\varepsilon/\beta)$ approximation to the optimal value. Here we used that $\mathsf{gap}(s) \geq \alpha s$.

The constant β equals the expected value of a predicate π on a random input for the worst predicate $\pi \in \Lambda$:

$$\beta = \min_{\pi \in \Lambda} \mathbb{E}_{x_1,\ldots,x_k \in D}[\pi(x_1, \ldots, x_k)].$$

The value of a random assignment is at least β, in expectation, for any instance of k-CSP Λ. Thus the optimal value of any maximization k-CSP Λ is at least β.

Similarly, Theorem 14 implies that no algorithm can achieve a better than an $(\alpha + O(\varepsilon))$ approximation if UGC holds. Indeed, consider an integrality gap instance $\mathcal{I}$ with the integrality gap $\alpha' \leq \alpha + \varepsilon$. Then, by Theorem 14, it is NP-hard to find a solution of value at least $\alpha'(\mathsf{SDP}+\varepsilon)+\varepsilon$ given an instance of value SDP. Thus no algorithm has an approximation factor better than

$$\frac{\alpha'(\mathsf{SDP} + \varepsilon) + \varepsilon}{\mathsf{SDP}} \leq \alpha' + \frac{2\varepsilon}{\mathsf{SDP}} \leq \alpha' + \frac{2\beta(\varepsilon/\beta)}{\mathsf{OPT}} \leq \alpha + \varepsilon + 2\varepsilon/\beta.$$

▶ **Corollary 16** (Raghavendra and Steurer [48], Raghavendra [47]). *For every maximization k-CSP Λ with nonnegative predicates and every positive ε, there exists a polynomial time $(\alpha - \varepsilon)$-approximation algorithm, where α is the integrality gap of the SDP relaxation from Section 6.1. Further, for every positive ε, there is no $(\alpha + \varepsilon)$-approximation algorithm if UGC holds.*

Note that for many maximization CSPs the best approximation ratio α is still not known.

From Theorems 14 and 15, we cannot get an analog of Corollary 16 for minimization versions of generalized CSPs with nonnegative predicates and for regular CSPs with objective (3) (described in the introduction). The reason for that is that the cost of a minimization CSP can be arbitrarily close to 0.[6] Similarly, we cannot get an analog of Corollary 16 for objective (2). What we can get is the following. Let $f(\delta) = 1 - \mathsf{gap}(1 - \delta)$. There exists an algorithm that given a $(1 - \delta)$-satisfiable instance and parameter $\varepsilon > 0$, finds an assignment satisfying a $(1 - f(\delta + \varepsilon) - \varepsilon)$ fraction of the constraints. The running time of the algorithm is polynomial in n and superexponential in $1/\varepsilon$. Note that typically we have to take $\delta = O(\varepsilon)$ to make this guarantee interesting. However, if we, say, let $\delta = \varepsilon$, we get an algorithm with running time superexponential in $1/\delta$.

6.1 SDP Relaxation for k-CSPs with $k > 2$

The SDP relaxation for CSPs of arity $k > 2$ consists of two parts: a semidefinite program and a linear program connected by special constraints. The SDP part has the same variables $\bar{u}_{ia}$ and $\bar{v}_0$ as the SDP relaxation for $k = 2$. These variables must satisfy SDP constraints (13) and (14).

The LP part has a variable $p_{i_1\ldots i_k}(a_1, \ldots, a_k) \in [0, 1]$ for every predicate $\pi(x_{i_1}, \ldots, x_{i_k})$ and $(a_1, \ldots, a_k) \in D^k$. For every predicate $\pi(x_{i_1}, \ldots, x_{i_k})$, we define a local probability

[6] In fact, most minimization CSPs studied in the literature – such as Min UnCut, Min 2CNF Deletion and Unique Games (with objective (3)) – do not admit a constant factor approximation if UGC holds. Their worst case instances have cost $o(1)$.

distribution on the assignments to the variables $x_{i_1}, \dots, x_{i_k}$:

$$\widetilde{\Pr}_{i_1,\dots,i_k}\big((x_{i_1},\dots,x_{i_k}) \in \mathcal{E}\big) = \sum_{(a_1,\dots,a_k)\in\mathcal{E}} p_{i_1\dots i_k}(a_1,\dots,a_k).$$

Formally, every $\widetilde{\Pr}_{i_1,\dots,i_k}$ is a linear combination of the LP variables $p_{i_1\dots i_k}(a_1,\dots,a_k)$. Now we can write the objective function as follows:

$$\frac{1}{|\Pi|}\sum_{\pi(x_{i_1},\dots,x_{i_k})\in\Pi}\ \sum_{(a_1,\dots,x_k)\in D^k} \pi(a_1,\dots,a_k)\widetilde{\Pr}_{i_1,\dots,i_k}\big(x_{i_1}=a_1,\dots,x_{i_k}=a_k\big).$$

We add an LP constraint $\widetilde{\Pr}_{i_1,\dots,i_k}\big((x_{i_1},\dots,x_{i_k})\in D^k\big)=1$. We then connect the LP and SDP by imposing the constraints

$$\widetilde{\Pr}_{i_1,\dots,i_k}(x_i=a) = \|\bar{u}_{ia}\|^2$$
$$\widetilde{\Pr}_{i_1,\dots,i_k}(x_i=a,\, x_j=b) = \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle$$

for all predicates $\pi(x_{i_1},\dots,x_{i_k})$ and $i,j\in\{i_1,\dots,i_k\}$. We refer the reader to the paper by Raghavendra [47] for more discussion on this SDP relaxation.

6.2 Rounding Algorithm for 2-CSPs with Nonnegative Predicates

The main observation behind the universal rounding algorithm is that for every SDP solution $\bar{u}_{ia}$ there exists another SDP solution $\bar{u}'_{ia}$ with a constant number (depending on ε and d) of distinct vectors $\bar{u}'_{ia}$ that has approximately the same SDP value as the original solution $\bar{u}_{ia}$. Here is the formal statement we need.

▶ **Theorem 17.** *For every positive ε, there exists a randomized polynomial time algorithm that given an instance $\mathcal{I}$ of a 2-CSP with the set of predicates Π and an SDP solution $\bar{u}_{ia}$ of value* SDP *returns a set of predicates Π' of size at least $(1-\varepsilon)|\Pi|$, an SDP solution $\bar{u}'_{ia}$ for the instance with predicates Π' of value at least $SDP-\varepsilon$, and a set $\mathcal{W}$ of size at most $f(\varepsilon,d)$ (for some function f depending only on ε and d) such that each tuple $w_i=(\bar{u}'_{i1},\dots,\bar{u}'_{id})$ belongs to $\mathcal{W}$. The algorithm fails with exponentially small probability.*

The proof of Theorem 17 is conceptually simple: To obtain vectors $\bar{u}'_{ia}$, we project the original vectors $\bar{u}_{ia}$ into a low dimensional space and round the projections to a fine epsilon net. The number of points in the net depends only on ε and d. The embedding into the net is almost isometric, and thus it almost preserves the SDP value. The main challenge is that the embedded vectors may violate the SDP constraints. So to make the solution feasible, we need to slightly perturb the embedded vectors. This step is rather technical and complicates the proof. We give a formal proof of Theorem 17 in Section 6.3. We now present and analyze the universal rounding algorithm.

The algorithm works in three phases: First, it solves the SDP relaxation and obtains a set of vectors $\bar{u}_{ia}$. Then, using Theorem 17, the algorithm transforms the SDP solution into another solution $\bar{u}'_{ia}$ of approximately the same value such that the number of distinct tuples $w_i=(\bar{u}'_{i1},\dots,\bar{u}'_{id})$ is upper bounded by some function $f(\varepsilon,d)$ of ε and d, which does not depend on the number of variables and constraints. We denote the set of all w_i by $\mathcal{W}$. The algorithm identifies variables x_i that are mapped to the same tuple of vectors w and obtains a new instance $\widetilde{\mathcal{I}}$ of 2-CSP Λ. Formally, the instance $\widetilde{\mathcal{I}}$ has a variable x'_w for each $w\in\mathcal{W}$ and a constraint $\pi(x'_{w_i},x'_{w_j})$ for each constraint $\pi(x_i,x_j)$ in $\mathcal{I}$. Note that $\widetilde{\mathcal{I}}$ has at

most $f(\varepsilon, d)$ variables. The algorithm finds the optimal solution x^*_w for $\widetilde{\mathcal{I}}$ by enumerating all $d^{|\mathcal{W}|}$ possible solutions and outputs the solution $x_i = x^*_{w_i}$ for the original instance $\mathcal{I}$.

Input: An instance $\mathcal{I}$ of 2-CSP Λ, parameter $\varepsilon > 0$.
Output: A solution x_i of value at least $\mathsf{gap}(\mathsf{SDP} - \varepsilon) - \varepsilon$.

1. Solve the SDP and obtain vectors $\bar{u}_{ia}$.
2. Transform vectors $\bar{u}_{ia}$ to vectors $\bar{u}'_{ia}$ and construct the set $\mathcal{W}$ using the algorithm from Theorem 17. Let $w_i = (\bar{u}'_{i1}, \ldots, \bar{u}'_{id})$ for each i. By Theorem 17, $w_i \in \mathcal{W}$.
3. Build a new instance $\widetilde{\mathcal{I}}$ of 2-CSP Λ: For each $w \in \mathcal{W}$, create a variable x'_w in $\widetilde{\mathcal{I}}$. For each constraint $\pi(x_i, x_j)$ between x_i and x_j in $\mathcal{I}$, add the constraint $\pi(x'_{w_i}, x'_{w_j})$ between x'_{w_i} and x'_{w_j} to $\widetilde{\mathcal{I}}$.
4. Find the optimal solution $x^*_{w_i}$ for $\widetilde{\mathcal{I}}$ by enumerating all possible solutions of $\widetilde{\mathcal{I}}$.
5. Output the solution $x_i = x^*_{w_i}$.

The running time of the algorithm is exponential in $|\mathcal{W}|$ but is polynomial in n, since the size of $\mathcal{W}$ is upper bounded by $f(\varepsilon, d)$ which depends only on ε and d. We show that the algorithm finds a solution of cost at least $\mathsf{gap}(\mathsf{SDP} - \varepsilon)$. Denote by $\mathcal{I}'$ and $\widetilde{\mathcal{I}}'$ subinstances of $\mathcal{I}$ and $\widetilde{\mathcal{I}}$ in which we removed predicates from $\Pi \setminus \Pi'$ and kept predicates from Π'. By Theorem 17, the value of the SDP solution $\bar{u}'_{ia}$ on the instance $\mathcal{I}'$ is at least $\mathsf{SDP} - \varepsilon$. Observe that the SDP solution $\bar{u}'_{ia}$ corresponds to a feasible SDP solution for the instance $\widetilde{\mathcal{I}}'$ in which the vectors for the variable x'_{w_i} are $\bar{u}'_{i1}, \ldots, \bar{u}'_{id}$. This SDP solution is well defined, since $(\bar{u}'_{i1}, \ldots, \bar{u}'_{id}) = w_i = w_j = (\bar{u}'_{j1}, \ldots, \bar{u}'_{jd})$ if $w_i = w_j$. The value of the SDP solution $\bar{u}'_{ia}$ on the instance $\mathcal{I}'$ equals the value of the corresponding SDP solution on $\widetilde{\mathcal{I}}'$ since for every constraint $\pi(x_i, x_j)$ in $\mathcal{I}'$, we have a constraint $\pi(x'_{w_i}, x'_{w_j})$ in $\widetilde{\mathcal{I}}'$, and the SDP value of the constraint π is the same in instances $\mathcal{I}'$ and $\widetilde{\mathcal{I}}'$: It is equal to $\sum_{ab} \pi(a, b)\langle \bar{u}'_{ia}, \bar{u}'_{jb}\rangle$. Since the optimal SDP value of the instance $\mathcal{I}'$ is at least $\mathsf{SDP} - \varepsilon$, the value of the optimal solution $x^*_{w_i}$ for $\mathcal{I}'$ is at least $\mathsf{gap}(\mathsf{SDP} - \varepsilon)$ (by the definition of $\mathsf{gap}(\cdot)$). The value of the solution $x_i = x^*_{w_i}$ for the instance $\mathcal{I}'$ is the same as the value of the solution $x^*_{w_i}$ for $\widetilde{\mathcal{I}}'$. If we omit the normalization factor $1/|\Pi|$ in the definition of the value of a solution, then the value of the solution x_i for the instance $\mathcal{I}$ will be greater than or equal to the value of the solution x_i for the instance $\mathcal{I}'$, since the value of removed predicates – predicates in $\Pi \setminus \Pi'$ – is nonnegative. With the normalization, we get that the value of the solution x_i for the original instance $\mathcal{I}$ is lower bounded by $(|\Pi'|/|\Pi|) \cdot \mathsf{gap}(\mathsf{SDP} - \varepsilon) \geq (1 - \varepsilon)\mathsf{gap}(\mathsf{SDP} - \varepsilon) \geq \mathsf{gap}(\mathsf{SDP} - \varepsilon) - \varepsilon$.

6.3 Proof of Theorem 17

Proof. We describe an algorithm for constructing the set $\mathcal{W}$ and vectors $\bar{u}'_{ia}$. The algorithm works in three steps: At the first step, it embeds the original SDP solution $\bar{u}_{ia}$ into a low dimensional space using the Johnson–Lindenstrauss transform [28]. Denote the embedding of the vector $\bar{u}_{ia}$ by $\varphi(\bar{u}_{ia})$. At the second step, the algorithm slightly perturbs vectors $\varphi(\bar{u}_{ia})$, so that the perturbed vectors $\varphi'(\bar{u}_{ia})$ satisfy all SDP constraints. At this step, the algorithm also removes some predicates from the set Π. Finally, at the third step, the algorithm picks an η-net (for sufficiently small η; $\eta = \varepsilon/C_d$) in the set of tuples $(\varphi'(\bar{u}_{i1}), \ldots, \varphi'(\bar{u}_{id}))$ equipped with the norm $\ell_2 \oplus_\infty \cdots \oplus_\infty \ell_2$ and for every i finds $w_i \in \mathcal{W}$ closest to $(\varphi'(\bar{u}_{i1}), \ldots, \varphi'(\bar{u}_{id}))$. It lets $\bar{u}'_{ia}$ be the a-th component of w_i. We show that for most variables x_i, x_j and values $a, b \in D$, $\langle \bar{u}'_{ia}, \bar{u}'_{jb}\rangle \approx \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle$, and hence the SDP value of the solution $\bar{u}'_{ia}$ approximately equals the SDP value of the optimal SDP solution $\bar{u}_{ia}$. We describe each step of the algorithm in detail below. We use the following notation in the proof: for every nonzero vector $\bar{u}$, let $\nu(\bar{u}) = \bar{u}/\|\bar{u}\|$. Let $\nu(0) = 0$.

Input: An SDP solution $\{\bar{u}_{ia}\}$ of value SDP and a parameter $\varepsilon \in (0,1)$.
Output: A set $\mathcal{W}$ of size at most $f(\varepsilon, d)$ and an SDP solution $\{\bar{u}'_{ia}\}$ of value at least $\mathsf{SDP} - \varepsilon$ such that $w_i = (\bar{u}'_{i1}, \dots, \bar{u}'_{id}) \in \mathcal{W}$ for each i.

1. Embed $\bar{u}_{ia}$ into a low dimensional space using the Johnson–Lindenstrauss transform as described in Section 6.3.1 and obtain vectors $\varphi'(\bar{u}_{ia})$.
2. Transform vectors $\varphi(\bar{u}_{ia})$ to vectors $\varphi'(\bar{u}_{ia})$ using the Gram–Schmidt process (see Lemma 22) and the procedure from Lemma 23 with $\mu = d^2(3^{d+1}\, d + 1)\eta$.
3. Find an η-net $\mathcal{W}$ in the set of all tuples $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$. For each i, let w_i be the vector in $\mathcal{W}$ closest to $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$ in the norm $\ell_2 \oplus_\infty \cdots \oplus_\infty \ell_2$. Let $\bar{u}'_{ia}$ be the a-th component of the tuple w_i.

6.3.1 Step I: Johnson–Lindenstrauss Transform

We use the Johnson–Lindenstrauss lemma and two simple corollaries which we state now.

▶ **Theorem 18** (Johnson–Lindenstrauss [28]). *For every $\eta \in (0,1)$ and $\delta \in (0,1)$, there exists an integer $m = O(\log(1/\delta)/\varepsilon^2)$ and a family Φ of linear operators from $\mathbb{R}^n$ to $\mathbb{R}^m$ such that for every $\bar{u} \in \mathbb{R}^n$ and a random $\varphi \in \Phi$, we have*

$$\Pr_{\varphi}(\|\bar{u}\|^2 \le \|\varphi(\bar{u})\|^2 \le (1+\eta)\|\bar{u}\|^2) \ge 1 - \delta.$$

Moreover, a random operator φ can be sampled from Φ in randomized polynomial time.

▶ **Corollary 19.** *For every $\eta \in (0,1)$, $\delta \in (0,1)$ and every unit vector $\bar{v}_0 \in \mathbb{R}^n$, there exists an integer $m = O(\log(1/\delta)/\varepsilon^2)$ and a family Φ of linear operators from $\mathbb{R}^n$ to $\mathbb{R}^m$ such that for every $\bar{u} \in \mathbb{R}^n$ and a random $\varphi \in \Phi$, we have $\|\varphi(\bar{v}_0)\| = 1$ a.s. and*

$$\Pr_{\varphi}((1-\eta)\|\bar{u}\|^2 \le \|\varphi(\bar{u})\|^2 \le (1+\eta)\|\bar{u}\|^2) \ge 1 - \delta. \tag{16}$$

Moreover, a random operator φ can be sampled from Φ in randomized polynomial time.

Proof. We simply let $\varphi(\bar{u}) = \tilde{\varphi}(\bar{u})/\|\tilde{\varphi}(\bar{v}_0)\|$, where $\tilde{\varphi}$ is the random operator from Theorem 18. ◀

▶ **Corollary 20.** *For a random $\varphi \in \Phi$ from Corollary 19 we have: for every $\bar{u}, \bar{v} \in \mathbb{R}^n$*

$$\Pr_{\varphi}(\langle \nu(\bar{u}), \nu(\bar{v})\rangle - 3\eta \le \langle \varphi(\nu(\bar{u})), \varphi(\nu(\bar{v}))\rangle \le \langle \nu(\bar{u}), \nu(\bar{v})\rangle + 3\eta) \ge 1 - 4\delta.$$

Proof. Consider φ from Corollary 19. Assume that φ preserves the lengths of vectors $\nu(\bar{u})$, $\nu(\bar{v})$, and $(\nu(\bar{u}) \pm \nu(\bar{v}))$ up to a factor $(1 \pm \eta)$ (as in Equation (16)). This happens with probability at least $1 - 4\delta$. We have

$$\begin{aligned}
2\langle \varphi(\nu(\bar{u})), \varphi(\nu(\bar{v}))\rangle &= \|\varphi(\nu(\bar{u})) + \varphi(\nu(\bar{v}))\|^2 - \|\varphi(\nu(\bar{u}))\|^2 - \|\varphi(\nu(\bar{v}))\|^2 \\
&\ge (1-\eta)\,\|\nu(\bar{u}) + \nu(\bar{v})\|^2 - (1+\eta)\,\|\nu(\bar{u})\|^2 - (1+\eta)\,\|\nu(\bar{v})\|^2 \\
&\ge \big(\|\nu(\bar{u}) + \nu(\bar{v})\|^2 - \|\nu(\bar{u})\|^2 - \|\nu(\bar{v})\|^2\big) - \\
&\quad - \eta\big(\|\nu(\bar{u}) + \nu(\bar{v})\|^2 + \|\nu(\bar{u})\|^2 + \|\nu(\bar{v})\|^2\big) \\
&\ge 2\langle \nu(\bar{u}), \nu(\bar{v})\rangle - 6\eta.
\end{aligned}$$

By applying the bound above to vectors $\bar{u}$ and $-\bar{v}$, we get $\langle \varphi(\nu(\bar{u})), \varphi(\nu(\bar{v}))\rangle \le \langle \nu(\bar{u}), \nu(\bar{v})\rangle + 3\eta$. ◀

At the first step, the algorithm embeds vectors $\bar{u}_{ia}$ into $m = O(\log(1/\delta')/\varepsilon^2)$ dimensional space for $\delta' = \eta/(8d^2)$ and $\eta' = \eta/6$ using Corollary 20. We show that vectors $\varphi(\bar{u}_{ia})$ satisfy the SDP constraint (13), almost satisfy SDP constraints (15), and almost preserve inner products between vectors.

▶ **Lemma 21.**

1. *For every $i \in [n]$, $\sum_{a\in D} \varphi(\bar{u}_{ia}) = \varphi(\bar{v}_0)$, and $\|\varphi(\bar{v}_0)\| = 1$ a.s.*
2. *For every $i \in [n]$,*

$$\Pr_{\varphi}\Big(|\langle \nu(\varphi(\bar{u}_{ia})), \nu(\varphi(\bar{u}_{ib}))\rangle| \le \eta \text{ for all } a \ne b\Big) \ge 1 - \eta.$$

3. *For all $i, j \in [n]$,*

$$\Pr_{\varphi}\Big(\langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \ge \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta \text{ for all } a, b\Big) \ge 1 - \eta.$$

Proof. Item 1 holds because φ is a linear operator and $\sum_{a\in D} \bar{u}_{ia} = \bar{v}_0$. Then, for every $a \ne b$, we have $\langle \bar{u}_{ia}, \bar{u}_{ib}\rangle = 0$, and hence $\langle \nu(\bar{u}_{ia}), \nu(\bar{u}_{ib})\rangle = 0$. By Corollary 20, $|\langle \varphi(\nu(\bar{u}_{ia})), \varphi(\nu(\bar{u}_{ib}))\rangle| \le \eta/2$ with probability at least $1 - \eta/d^2$ for each $a \ne b$, and, by Corollary 19, $\|\varphi(\nu(\bar{u}_{ia}))\| \ge 1 - \eta/6$ with probability $1 - \eta/(4d^2)$ for each a. Thus, for every $i \in V$, with probability at least $1 - \eta$, we have for all a, b:

$$|\langle \nu(\varphi(\bar{u}_{ia})), \nu(\varphi(\bar{u}_{ib}))\rangle| = \frac{|\langle \varphi(\nu(\bar{u}_{ia})), \varphi(\nu(\bar{u}_{ib}))\rangle|}{\|\varphi(\nu(\bar{u}_{ia}))\|\,\|\varphi(\nu(\bar{u}_{ia}))\|} \le \frac{\eta/2}{(1-\eta/6)^2} < \eta.$$

Hence, item 2 holds. Similarly, for every $i, j \in V$, with probability at least $1 - \eta$, we have for all a, b,

$$\langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \ge \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta\|\bar{u}_{ia}\|\|\bar{u}_{jb}\|/2 \ge \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta/2,$$

and $\|\varphi(\bar{u}_{ia})\| \le (1 + \eta/6)\|\bar{u}_{ia}\|$, $\|\varphi(\bar{u}_{jb})\| \le (1 + \eta/6)\|\bar{u}_{jb}\|$. Consequently,

$$\langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \ge \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta$$

for all a, b with probability $1 - \eta$. Hence, item 3 holds. ◀

6.3.2 Step II: Fixing Violated SDP Constraints

Lemma 21 shows that vectors $\varphi(\bar{u}_{ia})$ satisfy SDP constraints (13) and almost satisfy constraints (14) and (15) as $\langle \bar{u}_{ia}, \bar{u}_{jb}\rangle \approx \langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle$ for most $i, j \in [n]$ and $a, b \in D$. At the second step, the algorithm slightly perturbs vectors $\varphi(\bar{u}_{ia})$ to fix all SDP constraints. First, the algorithm applies the Gram–Schmidt orthogonalization process (described in Lemma 22 below) to vectors $\varphi(\bar{u}_{i1}), \dots, \varphi(\bar{u}_{id})$ for each $i \in [n]$ and obtains vectors $\varphi^{\perp}(\bar{u}_{i1}), \dots, \varphi^{\perp}(\bar{u}_{id})$ that satisfy constraints (14). Then, the algorithm transforms vectors $\varphi^{\perp}(\bar{u}_{ia})$ into vectors $\varphi'(\bar{u}_{ia})$ using the procedure from Lemma 23 with $\mu = d^2(3^{d+1}\,d + 1)\eta$. This fixes most constraints (15). The algorithm removes those predicates $\pi \in \Pi$ for which the constraint (15) is still violated. We denote obtained vectors by $\varphi'(\bar{u}_{ia})$ and the set of remained predicates by Π'. We now state Lemma 22 and Lemma 23.

▶ **Lemma 22** (Gram–Schmidt process)**.** *There exists a polynomial time algorithm that given vectors $\bar{v}_1, \dots, \bar{v}_d$ of length at most 1.5 returns vectors $\bar{v}'_1, \dots, \bar{v}'_d$ such that (1) $\langle \bar{v}'_a, \bar{v}'_b\rangle = 0$ for $a \ne b$, (2) $\sum_a \bar{v}_a = \sum_a \bar{v}'_a$, and (3) $\|\bar{v}'_a - \bar{v}_a\| \le 3^d d\eta$, where $\eta = \max_{a,b} |\langle \nu(\bar{v}_a), \nu(\bar{v}_b)\rangle|$.*

▶ **Lemma 23.** *There exists a polynomial-time algorithm that transforms any set of vectors $\bar{v}_{ia}$ satisfying the SDP constraints (13) and (14) into a set of vectors $\bar{v}'_{ia}$ also satisfying SDP constraints (13) and (14) for some unit vector $\bar{v}'_0$ such that $\langle \bar{v}'_{ia}, \bar{v}'_{jb}\rangle = (1-\mu)\langle \bar{v}_{ia}, \bar{v}_{jb}\rangle + \mu/d^2$ for all $i, j \in [n]$, $i \neq j$, $a, b \in D$, where $\mu \in [0,1]$ is a parameter.*

We first show that vectors $\varphi'(\bar{u}_{ia})$ satisfy all SDP constraints for the instance with predicates Π' and estimate the value of this SDP solution. Then, we prove Lemmas 22 and 23.

▶ **Lemma 24.** *Vectors $\varphi'(\bar{u}_{ia})$ together with the vector $\bar{v}'_0 = \varphi'(\bar{v}_0)$ satisfy all SDP constraints for the set of predicates Π'. Further the expected value of the SDP solution $\varphi'(\bar{u}_{ia})$ is at least $(1-2\mu)\mathsf{SDP}$, where SDP is the value of the solution $\bar{u}_{ia}$. The expectation is taken over a random embedding $\varphi \in \Phi$.*

Proof of Lemma 24. First, observe that vectors $\varphi'(\bar{u}_{ia})$ satisfy all SDP constraints (15), simply because we remove all predicates $\pi(x_i, x_j)$ for which these constraints are violated. By Lemma 23, vectors $\varphi'(\bar{u}_{ia})$ satisfy SDP constraints (13) and (14) if vectors $\varphi^{\perp}(\bar{u}_{ia})$ satisfy these constraints. By Lemma 21, item 1, we have $\sum_{a\in D} \varphi(\bar{u}_{ia}) = \varphi(\bar{v}_0)$ for all i. The Gram–Schmidt process preserves all sums $\sum_{a\in D} \varphi(\bar{u}_{ia})$ (by Lemma 22, item 2). Hence, vectors $\varphi^{\perp}(\bar{u}_{ia})$ satisfy SDP constraints (13). For every i, the Gram–Schmidt process transforms vectors $\bar{u}_{i1}, \dots, \bar{u}_{id}$ into orthogonal vectors; thus, $\varphi^{\perp}(\bar{u}_{ia})$ satisfy SDP constraints (14). This shows that vectors $\varphi'(\bar{u}_{ia})$ form a feasible SDP solution.

We now estimate the SDP value of the solution $\varphi'(\bar{u}_{ia})$. Let

$$V_\eta = \{i \in [n] : |\langle \nu(\varphi(\bar{u}_{ia})), \nu(\varphi(\bar{u}_{ib}))\rangle| \le \eta \text{ for } a \neq b; \text{ and } \|\varphi(\bar{u}_{ia})\|^2 \le 1.5 \text{ for all } a\};$$
$$\Pi_\eta = \{\pi(x_i, x_j) \in \Pi : \langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \ge \langle \bar{u}_{ia}, \bar{u}_{ib}\rangle - \eta \text{ for } a, b \in D; i, j \in V_\eta\}.$$

By Lemma 21, the sets V_η and Π_η contain almost all variables and predicates, respectively. Specifically, for every i, $\Pr(i \in V_\eta) \ge 1-2\eta$; for all $\pi(x_i, x_j) \in \Pi$, $\Pr(\pi(x_i, x_j) \in \Pi_\eta) \ge 1-5\eta$. We show that for all $\pi(x_i, x_j) \in \Pi_\eta$, SDP constraints (15) are satisfied, and, thus, $\Pi_\eta \subset \Pi'$. We use the following simple claim.

▶ **Claim 25.** *Consider four vectors $\bar{v}_1, \bar{v}_2$ and $\bar{v}'_1, \bar{v}'_2$. Suppose that $\|\bar{v}_1 - \bar{v}'_1\| \le \eta$, $\|\bar{v}_2 - \bar{v}'_2\| \le \eta$, and $\|\bar{v}'_1\|, \|\bar{v}'_2\| \le 1$ for some positive $\eta < 1$, then $\langle \bar{v}'_1, \bar{v}'_2\rangle \ge \langle \bar{v}_1, \bar{v}_2\rangle - 3\eta$.*

Proof. We have $\langle \bar{v}_1, \bar{v}_2\rangle = \big\langle \bar{v}'_1 - (\bar{v}'_1 - \bar{v}_1), \bar{v}'_2 - (\bar{v}'_2 - \bar{v}_2)\big\rangle \ge \langle \bar{v}'_1, \bar{v}'_2\rangle - \|\bar{v}'_1 - \bar{v}_1\|\,\|\bar{v}'_2\| - \|\bar{v}'_1\|\,\|\bar{v}'_2 - \bar{v}_2\| - \|\bar{v}'_1 - \bar{v}_1\|\|\bar{v}'_2 - \bar{v}_2\| \ge \langle \bar{v}'_1, \bar{v}'_2\rangle - 3\eta$. ◀

By the definition of Π_η, we have $\langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle \ge \langle \bar{u}_{ia}, \bar{u}_{jb}\rangle - \eta$. By Lemma 22, item 3, and Claim 25, we have $\langle \varphi^{\perp}(\bar{u}_{ia}), \varphi^{\perp}(\bar{u}_{jb})\rangle \ge \langle \varphi(\bar{u}_{ia}), \varphi(\bar{u}_{jb})\rangle - 3^d d\eta$. Then, by Lemma 23, $\langle \varphi'(\bar{u}_{ia}), \varphi'(\bar{u}_{jb})\rangle = (1-\mu)\langle \varphi^{\perp}(\bar{u}_{ia}), \varphi^{\perp}(\bar{u}_{jb})\rangle + \mu/d^2$. Putting these inequalities together, we get

$$\langle \varphi'(\bar{u}_{ia}), \varphi'(\bar{u}_{jb})\rangle \ge (1-\mu)\langle \bar{u}_{ia}, \bar{u}_{ib}\rangle - (3^{d+1}\,d + 1)\eta + \mu/d^2 = (1-\mu)\langle \bar{u}_{ia}, \bar{u}_{ib}\rangle. \tag{17}$$

Here, we used that $\mu/d^2 = (3^{d+1}\,d + 1)\eta$. Equation (17) implies that SDP constraints (15) are satisfied for vectors $\varphi'(\bar{u}_{ia})$.

We now estimate the value of the SDP solution. For every predicate $\pi(x_i, x_j)$ in Π_η, we have

$$\sum_{a,b\in D} \pi(a,b)\langle \varphi'(\bar{u}_{ia}), \varphi'(\bar{u}_{jb})\rangle \ge (1-\mu) \sum_{a,b\in D} \pi(a,b)\langle \bar{u}_{ia}, \bar{u}_{jb}\rangle.$$

Every predicate $\pi(x_i, x_j)$ in Π belongs to Π' with probability at least $(1-5\eta)$. Thus the expected SDP value for vectors $\varphi'(\bar{u}_{ia})$ is at least $(|\Pi|/|\Pi'|)\,(1-5\eta)(1-\mu)\mathsf{SDP} \ge (1-2\mu)\mathsf{SDP}$.

The multiplicative factor $(|\Pi|/|\Pi'|) \geq 1$ is due to the normalization factor $1/|\Pi|$ in the SDP objective. This finishes the proof of Lemma 24. ◀

Proof of Lemma 22. In the proof, we denote the projection of a vector $\bar{v}$ to a non-zero vector $\bar{u}$ by $\mathrm{proj}_{\bar{u}}\,\bar{v}$:

$$\mathrm{proj}_{\bar{u}}\,\bar{v} = \frac{\langle \bar{u}, \bar{v}\rangle}{\|\bar{u}\|^2}\,\bar{u} = \langle \bar{v}, \nu(\bar{u})\rangle \nu(\bar{u}).$$

As before $\nu(\bar{u}) = \bar{u}/\|\bar{u}\|$ for $\bar{u} \neq 0$. Observe that $\mathrm{proj}_{\bar{u}}\,\bar{v}$ is collinear with $\bar{u}$; and $(\bar{v} - \mathrm{proj}_{\bar{u}}\,\bar{v})$ is orthogonal to $\bar{u}$.

We describe an algorithm that transforms vectors $\bar{v}_1, \ldots \bar{v}_d$ to orthogonal vectors $\bar{v}'_1, \ldots, \bar{v}'_d$. The algorithm works in d iterations. After iteration t, it obtains a set of vectors $\bar{v}_1(t), \ldots, \bar{v}_d(t)$ satisfying the following properties: (1′) $\langle \bar{v}_a(t), \bar{v}_b(t)\rangle = 0$ for $a, b \leq t$, and $a \neq b$, (2′) $\sum_a \bar{v}_a(t) = \sum_a \bar{v}_a$, (3′) $\|\nu(\bar{v}_a(t)) - \nu(\bar{v}_a)\| \leq 3^a\eta$ for $a \leq t$ and $\bar{v}_a(t) = \bar{v}_a$ for $a > t$, and (4′) $\|\bar{v}_a(t) - \bar{v}_a\| \leq 3^t\eta$ for all a. The algorithm returns vectors $\bar{v}'_a = \bar{v}_a(d)$. Note that the desired conditions (1)–(3) follow from the conditions (1′)–(4′) on vectors $\bar{v}_a(d)$. At the first iteration, we set $\bar{v}_a(1) = \bar{v}_a$ for all a. At iteration $t \geq 2$, we let

$$\bar{v}_t(t) = \bar{v}_t(t-1) - \sum_{a<t} \mathrm{proj}_{\bar{v}_a(t-1)}\,\bar{v}_t(t-1); \tag{18}$$

$$\bar{v}_b(t) = \bar{v}_b(t-1) + \mathrm{proj}_{\bar{v}_b(t-1)}\big(\bar{v}_t(t-1) - \bar{v}_t(t)\big) \qquad \text{for } b < t; \tag{19}$$

$$\bar{v}_b(t) = \bar{v}_t(t-1) \qquad \text{for } b > t. \tag{20}$$

We prove by induction that properties (1′)–(4′) hold. It is easy to see that the properties hold for vectors $\bar{v}_a(0)$, since $\bar{v}_a(0) = \bar{v}_a$. Consider $t > 1$. Observe that vectors $\bar{v}_a(t)$ are collinear with vectors $\bar{v}_a(t-1)$ for $a < t$ and $\bar{v}_a(t) = \bar{v}_a(t-1)$ for $a > t$. The only vector that changes the direction is the vector $\bar{v}_t(t)$. The sum $\sum_{a<t} \mathrm{proj}_{\bar{v}_a(t-1)}\,\bar{v}_t(t-1)$ equals the projection of the vector $\bar{v}_t(t-1)$ to the span of orthogonal vectors $\bar{v}_1(t-1), \ldots, \bar{v}_{t-1}(t-1)$. Hence, $\bar{v}_t(t)$ is orthogonal to $\bar{v}_1(t-1), \ldots, \bar{v}_{t-1}(t-1)$, and, also, to vectors $v_1(t), \ldots, \bar{v}_{t-1}(t)$, which are collinear with $\bar{v}_1(t-1), \ldots, \bar{v}_{t-1}(t-1)$. Thus, property (1′) holds for t. The sum of vectors on the left hand side of (18–20) equals the sum of vectors on the right hand side of (18–20). Thus, property (2′) holds. For all $a \neq t$, we have $\nu(\bar{v}_a(t)) = \nu(\bar{v}_a(t-1))$. So we need to check (3′) only for $a = t$. Since $\bar{v}_t(t-1) = v_t$ and $\nu(\bar{v}_t(t-1)) = \nu(\bar{v}_t)$, we get

$$\begin{aligned}
\|\bar{v}_t(t) - \bar{v}_t\| &= \|\sum_{b<t} \mathrm{proj}_{\bar{v}_b(t)}\,\bar{v}_t\| = \|\sum_{b<t} \langle \nu(\bar{v}_b(t)), \bar{v}_t\rangle\, \nu(\bar{v}_b(t))\| \leq \sum_{b<t} |\langle \nu(\bar{v}_b(t)), \bar{v}_t\rangle| \\
&= \sum_{b<t} \big(|\langle \nu(\bar{v}_b), \bar{v}_t\rangle| + |\langle \nu(\bar{v}_b(t)) - \nu(\bar{v}_b), \bar{v}_t\rangle|\big) \\
&\leq \sum_{b<t} \big(|\langle \nu(\bar{v}_b), \nu(\bar{v}_t)\rangle| \cdot \|\bar{v}_t\| + \|\nu(\bar{v}_b(t)) - \nu(\bar{v}_b)\|\, \|\bar{v}_t\|\|\big) \\
&\leq \|\bar{v}_t\| \cdot \sum_{b<t} \big(|\langle \nu(\bar{v}_b), \nu(\bar{v}_t)\rangle| + \|\nu(\bar{v}_b(t)) - \nu(\bar{v}_b)\|\big).
\end{aligned}$$

We now upper bound $|\langle \nu(\bar{v}_b), \nu(\bar{v}_t)\rangle| \leq \eta$ and $\|\nu(\bar{v}_b(t)) - \nu(\bar{v}_b)\| \leq 3^b\eta$ and obtain the following inequality:

$$\|\bar{v}_t(t) - \bar{v}_t\| \leq \sum_{b<t} (\eta + 3^b\eta)\|\bar{v}_t\| = \big((t-1) + (3^t - 3)/2\big)\eta\|\bar{v}_t\| \leq 0.6 \cdot 3^t\eta\|\bar{v}_t\|. \tag{21}$$

The last inequality can be easily verified numerically. Let α be the angle between $\bar{v}_t(t)$ and $\bar{v}_t$. The formula above shows that $\sin(\alpha) = \|\bar{v}_t(t) - \bar{v}_t\|/\|\bar{v}_t\| \leq 0.6 \cdot 3^d\eta$. Observe that

$\|\nu(\bar{v}_t(t)) - \nu(\bar{v}_t)\| = 2\sin(\alpha/2)$ and $2\sin(\alpha/2) = \sin(\alpha)/\cos(\alpha/2) < 3^d$, since $\cos(\alpha/2) > 0.6$, if $\sin\alpha < 1/3$. This proves property $(3')$.

Finally, property $(4')$ holds, since for $a < t$, $\|\bar{v}_a(t) - \bar{v}_a(t-1)\| = |\langle \bar{v}_t(t) - \bar{v}_t(t-1), \nu(\bar{v}_a(t-1))\rangle| \le 3^t\eta$ by (21); $\|\bar{v}_t(t) - \bar{v}_t(t-1)\| \le 3^t\eta$ also by (21); and for $a > t$, $\bar{v}_a(t) = \bar{v}_a(t-1)$. ◀

Proof of Lemma 23. Observe that it is sufficient to prove Lemma 23 for $\mu^* = 1$: If vectors $\bar{z}_{ia}$ satisfy the conditions of this lemma for $\mu^* = 1$, then vectors $\bar{v}'_{ia} = \sqrt{1-\mu}\,\bar{v}_{ia} \oplus \sqrt{\mu}\,\bar{z}_{ia}$ satisfy the conditions of the lemma for any μ (with $\bar{v}'_0 = \sum_a \bar{v}'_{ia}$, which does not depend on i). Here $\oplus$ denotes the direct sum. This follows from the following identity: $\langle \bar{v}'_{ia}, \bar{v}'_{jb}\rangle = (1-\mu)\langle \bar{v}_{ia}, \bar{v}_{jb}\rangle + \mu\langle \bar{z}_{ia}, \bar{z}_{jb}\rangle$.

To construct vectors $\bar{z}_{ia}$, consider the random assignment algorithm. The algorithm independently assigns a random value from D to every variable x_i. Let x^{rnd}_{ia} be the indicator random variable of the event that the algorithm sets $x_i = a$. The random variables x^{rnd}_{ia} lie in the L_2 space equipped with the standard inner product $\langle x^{rnd}_{ia}, x^{rnd}_{jb}\rangle = \mathbb{E}[x^{rnd}_{ia}\, x^{rnd}_{jb}]$. It easy to see that $\|x^{rnd}_{ia}\|^2 = 1/d$, $\langle x^{rnd}_{ia}, x^{rnd}_{jb}\rangle = 1/d^2$ for $i \neq j$, $\langle x^{rnd}_{ia}, x^{rnd}_{ib}\rangle = 0$ for $a \neq b$. Furthermore, $\sum_a x^{rnd}_{ia} = 1$ for every i. We isometrically embed x^{rnd}_{ia} from L_2 into ℓ_2 and obtain vectors $\bar{z}_{ia}$. ◀

6.3.3 Step III: Rounding to a η-Net

At the last step, the algorithm picks an η-net in the set of all tuples $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$ equipped with the norm $\ell_2 \oplus_\infty \cdots \oplus_\infty \ell_2$. In this norm, the distance between two tuples $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$ and $(\varphi'(\bar{u}_{j1}), \dots, \varphi'(\bar{u}_{jd}))$ equals $\max_{a\in D} \|\varphi'(\bar{u}_{ia}) - \varphi'(\bar{u}_{ja})\|_2$. We denote the η-net by $\mathcal{W}$. The size of an η-net in the m-dimensional space, where all vectors $\varphi'(\bar{u}_{ia})$ lie, is upper bounded by $(1 + 2/\eta)^m$. Thus the size of $\mathcal{W}$ is upper bounded by $(1 + 2/\eta)^{md}$. This number, $(1 + 2/\eta)^{md}$, depends on m, d and η, which in turn depend only on ε and d. For each $i \in [n]$, the algorithm picks $w_i \in \mathcal{W}$ closest to $(\varphi'(\bar{u}_{i1}), \dots, \varphi'(\bar{u}_{id}))$, sets $\bar{u}'_{ia}$ to be the a-th component of w_i, and outputs all vectors $\bar{u}'_{ia}$.

Observe that for each i there is a j such that $\bar{u}'_{ia} = \varphi'(\bar{u}_{ja})$ for all a. Thus, vectors $\bar{u}'_{ia}$ satisfy all SDP constraints. We need to lower bound the SDP value of the solution $\bar{u}'_{ia}$. Note that for each i and a, $\|\bar{u}_{ia} - \varphi'(\bar{u}_{ia})\| \le \eta$, since $\mathcal{W}$ is an η-net. By Claim 25, $\langle \bar{u}'_{ia}, \bar{u}'_{jb}\rangle \ge \langle \varphi'(\bar{u}_{ia}), \varphi'(\bar{u}_{jb})\rangle - 3\eta$. Thus, the value of the SDP solution u'_{ia} is at least the value of the SDP solution for $\varphi'(\bar{u}_{ia})$ minus $3d^2\eta$. Thus, it is lower bounded by $(1-2\mu)\mathsf{SDP} - 3d^2\eta \ge \mathsf{SDP} - d^2(3^{d+1}\,d + 4)\eta$, where SDP is the SDP value of the solution $\bar{u}_{ia}$. This finishes the proof of Theorem 18. ◀

7 Open Problems

The most important open problem in the field is to prove or disprove the Unique Games Conjecture. Here, we list some other interesting open problems.

▶ Open Problem 1. Close the gap between the known approximation factors and hardness results for the following problems: Max 2-And, Max SAT, Multiway Cut.

▶ Open Problem 2. We know that the best possible approximation factor for Max k-And and all Boolean k-CSPs is $c_k k/2^k$, where $c_k = \Theta(1)$. Determine if the limit $\lim_{k\to\infty} c_k$ exists. If it does, find it.

Austrin and Mossel [7] proved that $c_k \le 1 + o(1)$ if UGC holds, and Chan [13] showed that $c_k \le 1 + o(1)$ for infinitely many k if $P \neq NP$. We conjecture that this upper bound is tight, $c_k = 1 \pm o(1)$, and, moreover, the algorithm from [43] has an approximation factor of $(1 - o(1))k/2^k$.

▶ Open Problem 3. Prove or disprove that the currently best known approximation factor of $\Omega(d \max(k, \log d)/d^k)$ for k-CSP(d) is asymptotically optimal. It is known that this approximation factor is optimal when $d = \Omega(k)$ [13].

▶ Open Problem 4. Suppose that the integrality gap of a minimization k-CSP Λ is α_n (α_n may depend on the number of variables n). Does there exist a polynomial-time algorithm with an approximation factor $(1 + \varepsilon)\alpha_n$ for every positive ε?

References

1 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for Min UnCut, Min 2CNF Deletion, and directed cut problems. In *Proceedings of the Symposium on Theory of Computing*, pages 573–581, 2005.

2 Haris Angelidakis, Yury Makarychev, and Pasin Manurangsi. An improved integrality gap for the călinescu–karloff–rabani relaxation for multiway cut. *arXiv preprint arXiv:1611.05530*, 2016.

3 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 563–572, 2010.

4 Sanjeev Arora, Rong Ge, and Ali Kemal Sinop. Towards a better approximation for sparsest cut? In *Proceedings of the Foundations of Computer Science*, pages 270–279, 2013.

5 Sanjeev Arora, Subhash A. Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K. Vishnoi. Unique games on expanding constraint graphs are easy. In *Proceedings of the Symposium on Theory of Computing*, pages 21–28, 2008.

6 Per Austrin. Towards sharp inapproximability for any 2-CSP. *SIAM Journal on Computing*, 39(6):2430–2463, 2010.

7 Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. *Computational Complexity*, 18(2):249–271, 2009.

8 Adi Avidor, Ido Berkovitch, and Uri Zwick. Improved approximation algorithms for Max NAE-SAT and Max SAT. In *Approximation and Online Algorithms*, pages 27–40. Springer, 2005.

9 Nikhil Bansal, Uriel Feige, Robert Krauthgamer, Konstantin Makarychev, Viswanath Nagarajan, Joseph Naor, and Roy Schwartz. Min-max graph partitioning and small set expansion. In *Proceedings of the Foundations of Computer Science*, pages 17–26, 2011.

10 Niv Buchbinder, Joseph Seffi Naor, and Roy Schwartz. Simplex partitioning via exponential clocks and the multiway cut problem. In *Proceedings of the Symposium on Theory of Computing*, pages 535–544, 2013.

11 Niv Buchbinder, Roy Schwartz, and Baruch Weizman. Simplex transformations and the multiway cut problem. In *Proceedings of the Symposium on Discrete Algorithms*, 2017. to appear.

12 Gruia Călinescu, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for multiway cut. In *Proceedings of the Symposium on Theory of Computing*, pages 48–52, 1998.

13 Siu On Chan. Approximation resistance from pairwise-independent subgroups. *J. ACM*, 63(3):27:1–27:32, August 2016.

14 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for unique games. In *Proceedings of the Symposium on Theory of Computing*, pages 205–214, 2006.

15 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Transactions on Algorithms (TALG)*, 5(3):32, 2009.

16 Eden Chlamtac, Konstantin Makarychev, and Yury Makarychev. How to play unique games using embeddings. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 687–696, 2006.

17 Miroslav Chlebík and Janka Chlebíková. On approximation hardness of the Minimum 2SAT-Deletion problem. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*, pages 263–273, 2004.

18 William H. Cunningham and Lawrence Tang. Optimal 3-terminal cuts and linear programming. In *Proceedings of the International Conference on Integer Programming and Combinatorial Optimization*, pages 114–125, 1999.

19 E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

20 Ari Freund and Howard Karloff. A lower bound of $8/(7+1/(k-1))$ on the integrality ratio of the călinescu–karloff–rabani relaxation for multiway cut. *Information Processing Letters*, 75(1):43–50, 2000.

21 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for Maximum Cut and Satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

22 Anupam Gupta and Kunal Talwar. Approximating unique games. In *Proceedings of the Symposium on Discrete Algorithm*, pages 99–106, 2006.

23 Venkatesan Guruswami, Johan Håstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. Beating the random ordering is hard: Every ordering CSP is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011.

24 Venkatesan Guruswami and Euiwoong Lee. Complexity of approximating CSP with balance/hard constraints. In *Proceedings of the Conference on Innovations in Theoretical Computer Science*, pages 439–448, 2014.

25 Venkatesan Guruswami and Yuan Zhou. Tight bounds on the approximability of almost-satisfiable horn sat and exact hitting set. In *Proceedings of the Symposium on Discrete Algorithms*, pages 1574–1589, 2011.

26 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

27 Johan Håstad, Sangxia Huang, Rajsekar Manokaran, Ryan O'Donnell, and John Wright. Improved NP-inapproximability for 2-variable linear equations. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 40. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

28 William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

29 David Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM (JACM)*, 45(2):246–265, 1998.

30 David R. Karger, Philip Klein, Cliff Stein, Mikkel Thorup, and Neal E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Mathematics of Operations Research*, 29(3):436–461, 2004.

31 Howard Karloff and Uri Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the Foundations of Computer Science*, pages 406–415, 1997.

32 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Symposium on Theory of Computing*, pages 767–775, 2002.

33 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007.

34 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. In *Proceedings of the Conference on Computational Complexity*, pages 379–386, 2003.

35 Guy Kindler, Alexandra Kolla, and Luca Trevisan. Approximation of non-boolean 2CSP. In *Proceedings of the Symposium on Discrete Algorithms*, pages 1705–1714, 2016.

36 Alexandra Kolla, Konstantin Makarychev, and Yury Makarychev. How to play unique games against a semi-random adversary: Study of semi-random models of unique games. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 443–452, 2011.

37 Robert Krauthgamer, Joseph Seffi Naor, and Roy Schwartz. Partitioning graphs into balanced components. In *Proceedings of the Symposium on Discrete Algorithms*, pages 942–949, 2009.

38 Michael Langberg, Yuval Rabani, and Chaitanya Swamy. Approximation algorithms for graph homomorphism problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 176–187. Springer, 2006.

39 Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *Proceedings of Integer Programming and Combinatorial Optimization*, pages 67–82. Springer, 2002.

40 Anand Louis and Konstantin Makarychev. Approximation algorithm for sparsest k-partitioning. In *Proceedings of the Symposium on Discrete Algorithms*, pages 1244–1255, 2014.

41 Anand Louis and Yury Makarychev. Approximation algorithms for hypergraph small-set expansion and small-set vertex expansion. *Theory of Computing*, 12(17):1–25, 2016.

42 Konstantin Makarychev and Yury Makarychev. How to play unique games on expanders. In *Approximation and Online Algorithms*, volume 6534 of *Lecture Notes in Computer Science*, pages 190–200. Springer Berlin / Heidelberg, 2011.

43 Konstantin Makarychev and Yury Makarychev. Approximation algorithm for non-Boolean Max k-CSP. *Theory of Computing*, 10(13):341–358, 2014.

44 Konstantin Makarychev and Yury Makarychev. Nonuniform graph partitioning with unrelated weights. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*, pages 812–822, 2014.

45 Rajsekar Manokaran, Joseph Seffi Naor, Prasad Raghavendra, and Roy Schwartz. Sdp gaps and ugc hardness for multiway cut, 0-extension, and metric labeling. In *Proceedings of the Symposium on Theory of Computing*, pages 11–20, 2008.

46 Pasin Manurangsi, Preetum Nakkiran, and Luca Trevisan. Near-optimal UGC-hardness of approximating Max k-CSP$_r$. In *Proceedings of the Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 15:1–15:28, 2016.

47 Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the Symposium on Theory of Computing*, pages 245–254, 2008.

48 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the Symposium on Theory of Computing*, pages 755–764, 2010.

49 Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the Symposium on Theory of Computing*, pages 191–199, 2000.

50 Ankit Sharma and Jan Vondrák. Multiway cut, pairwise realizable distributions, and descending thresholds. In *Proceedings of the Symposium on Theory of Computing*, pages 724–733, 2014.

51 Luca Trevisan. Approximation algorithms for unique games. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 197–205, 2005.

52 Jiawei Zhang, Yinyu Ye, and Qiaoming Han. Improved approximations for Max set splitting and Max NAE SAT. *Discrete Applied Mathematics*, 142(1):133–149, 2004.

53 Uri Zwick. Finding almost-satisfying assignments. In *Proceedings of the Symposium on Theory of Computing*, pages 551–560, 1998.

54 Uri Zwick. Computer assisted proof of optimal approximability results. In *Proceedings of the Symposium on Discrete Algorithms*, pages 496–505, 2002.

A Estimates for Gaussian Distribution

In this section, we prove several useful estimates on the Gaussian distribution. Let $X \sim \mathcal{N}(0,1)$ be one dimensional Gaussian random variable. Denote the probability that $X \geq t$ by $\bar{\Phi}(t)$:

$$\bar{\Phi}(t) = \Pr(X \geq t).$$

The first lemma gives a very accurate estimate on $\bar{\Phi}(t)$ for large t.

▶ **Lemma 26.** *For every $t > 0$,*

$$\frac{t}{\sqrt{2\pi}\,(t^2+1)} e^{-\frac{t^2}{2}} < \bar{\Phi}(t) < \frac{1}{\sqrt{2\pi}\,t} e^{-\frac{t^2}{2}}.$$

Proof. Write

$$\begin{aligned}\bar{\Phi}(t) &= \frac{1}{\sqrt{2\pi}} \int_t^\infty e^{-\frac{x^2}{2}} dx = \frac{1}{\sqrt{2\pi}} \left[\frac{-e^{-\frac{x^2}{2}}}{x} \Big|_t^\infty - \int_t^\infty \frac{e^{-\frac{x^2}{2}}}{x^2} dx \right] \\ &= \frac{1}{\sqrt{2\pi}t} e^{-\frac{t^2}{2}} - \frac{1}{\sqrt{2\pi}} \int_t^\infty \frac{e^{-\frac{x^2}{2}}}{x^2} dx.\end{aligned}$$

Thus,

$$\bar{\Phi}(t) < \frac{1}{\sqrt{2\pi}t} e^{-\frac{t^2}{2}}.$$

On the other hand,

$$\frac{1}{\sqrt{2\pi}} \int_t^\infty \frac{e^{-\frac{x^2}{2}}}{x^2} dx < \frac{1}{\sqrt{2\pi}t^2} \int_t^\infty e^{-\frac{x^2}{2}} dx = \frac{\bar{\Phi}(t)}{t^2}.$$

Hence,

$$\bar{\Phi}(t) > \frac{1}{\sqrt{2\pi}t} e^{-\frac{t^2}{2}} - \frac{\bar{\Phi}(t)}{t^2},$$

and, consequently,

$$\bar{\Phi}(t) > \frac{t}{\sqrt{2\pi}(t^2+1)} e^{-\frac{t^2}{2}}.$$

◀

▶ **Lemma 27.** *Let X and Y be Gaussian $\mathcal{N}(0,1)$ random variables with covariance $\operatorname{cov}(X,Y) = 1 - 2\varepsilon^2$. Pick the threshold $t > 1$ such that $\bar{\Phi}(t) = 1/L$ for $L > 3$. Then*

$$\Pr(X \geq t \text{ and } Y \leq t) = O(\varepsilon\sqrt{\log L}/L).$$

Proof. If $\varepsilon t \geq 1$ or $\varepsilon \geq 1/2$, then we are done, since $\varepsilon\sqrt{\log L} = \Omega(\varepsilon t) = \Omega(1)$ and

$$\Pr(X \geq t \text{ and } Y \leq t) \leq \Pr(X \geq t) = \frac{1}{L}.$$

So we assume that $\varepsilon t \leq 1$ and $\varepsilon < 1/2$. Let

$$\xi = \frac{X+Y}{2\sqrt{1-\varepsilon^2}}; \qquad \eta = \frac{X-Y}{2\varepsilon}.$$

Note that ξ and η are $\mathcal{N}(0,1)$ Gaussian random variables with covariance 0. Hence, ξ and η are independent. We have

$$\Pr\big(X \geq t \text{ and } Y \leq t\big) = \Pr\big(\sqrt{1-\varepsilon^2}\,\xi + \varepsilon\eta \geq t \text{ and } \sqrt{1-\varepsilon^2}\,\xi - \varepsilon\eta \leq t\big).$$

Denote by $\mathcal{E}$ the following event:

$$\mathcal{E} = \big\{\sqrt{1-\varepsilon^2}\,\xi + \varepsilon\eta \geq t \text{ and } \sqrt{1-\varepsilon^2}\,\xi - \varepsilon\eta \leq t\big\}.$$

Then,

$$\Pr\big(X \geq t \text{ and } Y \leq t\big) = \Pr(\mathcal{E} \text{ and } \varepsilon\eta \leq t) + \Pr(\mathcal{E} \text{ and } \varepsilon\eta \geq t).$$

Observe that the second probability on the right hand side is very small. It is upper bounded by $\Pr(\varepsilon\eta \geq t)$, which, in turn, is bounded as follows:

$$\Pr(\varepsilon\eta \geq t) = \frac{1}{\sqrt{2\pi}} \int_{t/\varepsilon}^{\infty} e^{-\frac{x^2}{2}}\,dx = \bar{\Phi}(t/\varepsilon) \leq O\Big(\frac{\varepsilon\, e^{-\frac{t^2}{2\varepsilon^2}}}{t}\Big) \leq O\Big(\frac{\varepsilon\, e^{-\frac{t^2}{2}}}{t}\Big) = O(\varepsilon/L).$$

We now estimate the first probability:

$$\begin{aligned}
\Pr(\mathcal{E} \text{ and } \varepsilon\eta \leq t) &= \mathbb{E}_\eta[\Pr(\mathcal{E} \text{ and } \eta \leq t/\varepsilon \mid \eta)] \\
&= \frac{1}{\sqrt{2\pi}} \int_0^{t/\varepsilon} \Pr(\mathcal{E} \mid \eta = x)\, e^{-x^2/2}\,dx \\
&= \frac{1}{\sqrt{2\pi}} \int_0^{t/\varepsilon} \Pr(\sqrt{1-\varepsilon^2}\xi \in [t-\varepsilon x, t+\varepsilon x])\, e^{-x^2/2}\,dx.
\end{aligned}$$

The density of the random variable $\sqrt{1-\varepsilon^2}\,\xi$ in the interval $(t-\varepsilon x, t+\varepsilon x)$ for $x \in [0, t/\varepsilon]$ is at most

$$\frac{1}{\sqrt{2\pi(1-\varepsilon^2)}} e^{\frac{-(t-\varepsilon x)^2}{2(1-\varepsilon^2)}} \leq \frac{1}{2}\, e^{\frac{-(t-\varepsilon x)^2}{2}} \leq \frac{1}{2}\, e^{\frac{-t^2}{2}} \cdot e^{\varepsilon t x} \leq \frac{1}{2}\, e^{\frac{-t^2}{2}} \cdot e^{x},$$

here we used that $\varepsilon \geq 1/2$ and $\varepsilon t \geq 1$. Hence,

$$\Pr(t - \varepsilon x \leq \sqrt{1-\varepsilon^2}\,\xi \leq t + \varepsilon x) \leq \varepsilon x\, e^{\frac{-t^2}{2}} \cdot e^{x}.$$

Therefore,

$$\Pr(\mathcal{E} \text{ and } \varepsilon\eta \leq t) \leq \frac{\varepsilon\, e^{\frac{-t^2}{2}}}{\sqrt{2\pi}} \int_0^{t/\varepsilon} x e^{x} \cdot e^{\frac{-x^2}{2}}\,dx \leq \frac{\varepsilon\, e^{\frac{-t^2}{2}}}{\sqrt{2\pi}} \underbrace{\int_0^{\infty} x e^{x} \cdot e^{\frac{-x^2}{2}}\,dx}_{O(1)}.$$

The integral in the right hand side does not depend on any parameters, so it can be upper bounded by some constant (e.g. one can show that it is upper bounded by $2\sqrt{2\pi}$). We get

$$\Pr(\mathcal{E} \text{ and } \varepsilon\eta \leq t) \leq O(\varepsilon\, e^{\frac{-t^2}{2}}) = O(\varepsilon \cdot t\bar{\Phi}(t)) = O(\varepsilon\sqrt{\log L}/L).$$

This finishes the proof. ◀