Approximating kCSP For Large Alphabets

Gil Goldshlager* and Dana Moshkovitz **

Massachusetts Institute of Technology

Abstract. In this work we show that the constraint satisfaction problem (CSP), where constraints depend on k variables each and variables range over alphabet of size d, can be efficiently approximated to within $\Omega(kd/d^k)$ for any k,d. Previous work by Makarychev and Makarychev obtained an approximation ratio of $\Omega(kd/d^k)$ only for the case where the alphabet size d is at most exponentially large in k. In contrast, much of the research on PCP focuses on the case of constant k and large d.

1 Introduction

Much in the same way that 3SAT is the canonical NP-Complete problem, kCSP is the canonical NP-hard approximation problem. In $k\text{CSP}_d$, one is given as input a set V of variables ranging over an alphabet of size d, as well as a family of constraints over the variables, where each constraint acts on k variables. The goal is to find an assignment to V that satisfies as many constraints as possible. For example, 3SAT is a special case of $k\text{CSP}_d$ where $k = 3$, $d = 2$, and the constraints are ORs of three literals each. Analogously to Cook's Theorem for 3SAT, the PCP Theorem [6, 5, 12, 2, 1] shows that there exists a constant k, such that for all $\varepsilon > 0$, for sufficiently large $d = d(\varepsilon)$, $k\text{CSP}_d$ is NP-hard to approximate to within ε. Specifically, it is NP-hard to distinguish the case where all the constraints can be simultaneously satisfied and the case where only ε fraction of the constraints can be simultaneously satisfied.

The theory of hardness of approximation builds on the PCP Theorem to prove, via appropriate reductions, tight hardness of approximation results. One is typically interested in PCP theorems with small ε, since the inapproximability factor one obtains building on the PCP theorem has an additive $\varepsilon^{\Omega(1)}$. To allow for a small ε, one considers a large alphabet size $d \geq (1/\varepsilon)^{\Omega(1)}$. To see why this is necessary, note that a random assignment to the variables satisfies d^{-k} fraction of the constraints in expectation, and hence one can efficiently find an assignment that satisfies $poly(1/d)$ fraction of the constraints,assuming $k = \Theta(1)$. Indeed, a large number of works in PCP prove such low error PCP theorems, e.g., [3, 17, 9, 16].

* ggoldsh@mit.edu. Department of Mathematics, MIT.

** dmoshkov@csail.mit.edu. Department of Electrical Engineering and Computer Science, MIT. This material is based upon work supported by the National Science Foundation under Grant Number 1218547.

The approximability of $k\text{CSP}_d$ was analyzed in numerous works. Often the first works focused on the Boolean case $d = 2$, and later works extended the results to more general d's. Samorodnitsky and Trevisan [18] showed that $k\text{CSP}_2$ is NP-hard to approximate to within $O(2^{2\sqrt{k}}/2^k)$. This was slightly improved to $O(2^{\sqrt{2k}}/2^k)$ by Engebretsen and Holmerin [11] and generalized by Engebretsen [10] to any constant $d \geq 2$, establishing a hardness result of $O(d^{O(\sqrt{k})}/d^k)$. Under the Unique Games Conjecture (UGC), better hardness results were obtained. Samorodnitsky and Trevisan [19] showed that $k\text{CSP}_2$ is UGC-hard to approximate to within $O(k/2^k)$. Austrin and Mossel [4] and Guruswami and Raghavendra [13] proved a UGC-hardness for $k\text{CSP}_d$ to within $O(kd^2/d^k)$ for any constants k, d. Moreover, Austrin and Mossel [4] improved this to $O(kd/d^k)$ for infinitely many k's. Håstad [15] proved that this hardness factor holds under the Unique Games Conjecture for all constants $k \geq d$. Chan [7] was able to prove the same hardness factor without relying on the Unique Games Conjecture. For the case of super-constant d and sufficiently large constant k, it is only known that $k\text{CSP}_d$ is NP-hard to approximate to within $O(d^{-1})$ [9]. We conjecture that the NP-hardness result for approximating within a factor of $O(kd/d^k)$ should extend to super-constant d, but current methods fail to prove strong bounds in this case due to their reliance on the long code.

There is also a long sequence of algorithmic results approximating $k\text{CSP}_d$. Trevisan [20] and Hast [14] studied the Boolean case. Charikar, Makarychev and Makarychev [8] settled the Boolean case up to constant factors, giving an efficient SDP-based approximation algorithm with approximation factor $\Omega(k/2^k)$. For the case of general d, Charikar et al [8] could only achieve a factor of $\Omega(k \log d/d^k)$. Makarychev and Makarychev [15] improved the approximation factor to $\Omega(kd/d^k)$ in the case $k \geq \Omega(\log d)$ (i.e., the alphabet is of size at most exponential in k). In the case that $k \leq O(\log d)$, their algorithm gives an approximation ratio of e^k/d^k, which is no better than the $\Omega(d/d^k)$ ratio that can be obtained by a simple greedy algorithm. To the best of our knowledge, there are no better approximation algorithms for the case of small k (possibly constant) and large alphabet, which is the setup that low error PCP Theorems focus on. The current work remedies this situation by presenting an approximation algorithm for $k\text{CSP}_d$ that achieves approximation ratio $\Omega(kd/d^k)$ for all values of k and d.

Theorem 1. *For all k, d, $k\text{CSP}_d$ can be efficiently approximated to within $\Omega(kd/d^k)$.*

Similarly to previous works on approximation of $k\text{CSP}_d$, we start by reducing the problem to a constraint satisfaction problem in which each constraint has a single satisfying assignment. Then we consider a natural semidefinite program (SDP) for the problem, and design a rounding algorithm that, given a vector solution, generates a discrete solution whose value is within $\Omega(kd/d^k)$ of the optimum. The SDP has a vector for each clause, as well as d vectors for each variable: one for every possible assignment to the variable.

The basic idea is to use the Goemans-Williamson rounding technique of projecting the vectors on a random Gaussian vector. First, we note that a random assignment yields the desired $\Omega(kd)$ gain as long as the clause vector is of magnitude (square l_2 norm) at most $O(1/kd)$. The Gaussian projection technique then suffices for clauses vectors of magnitude at least $\Omega(1/kd)$ and variable vectors of magnitude at most $O(1/d)$. Makarychev and Makarychev build on this to design an algorithm that handles vectors of magnitude possibly $\Omega(1/d)$, but only in the case of small d.

The contribution of the current paper is to handle the case of vectors of magnitude $\Omega(1/d)$ for all d. By the design of the SDP, there cannot be many vectors of magnitude much larger than $1/d$. However, it is the case with many vectors of magnitude only somewhat larger than $1/d$ that is the difficult case. When d is small, this distinction does not influence the approximation factor much, and this is what Makarychev and Makarychev use. However, when d is large, the approximation ratio that Makarychev and Makarychev obtain is significantly lower than it should be. The current paper handles the case of large d by partitioning the space of possible magnitudes into very carefully chosen intervals, and focusing on one interval which contains many vectors. The intervals are picked using a recursive formula which guarantees that one of them must contain enough vectors to contribute sufficiently many satisfied clauses to the rounded solution.

An interesting point is how our algorithm functions in the case of small d, when the solution of Makarychev and Makarychev also suffices. In fact, in this case, our algorithm almost degenerates into their algorithm. This is because the number of intervals chosen by our algorithm depends on the parameters k and d such that, in this small d case, there is only one interval. This point is further highlighted by Remark 1 of Section 4.

1.1 Organization

In Section 2, we define some useful notation for dealing with the k–CSP_d problem, as well as the semidefinite program (SDP) that we use. In Section 3, we present our algorithm for approximating k–CSP_d. In Section 4, we restate our main result, Theorem 1. We then state and prove three auxiliary theorems, and finally use these theorems to prove Theorem 1. For the sake of brevity, we defer the proofs of several technical lemmas to Section 5. Also, since the last of the auxiliary theorems is only a restructuring of the work of Makarychev and Makarychev [15], its proof is postponed to Section 6 along with the description of the algorithm it applies to. We present in Section 7 the details of a greedy algorithm that we use for the case of constant k. Finally, in Section 8, we describe how to reduce any k–CSP_d instance to an equivalent instance with only k-AND clauses.

2 Preliminaries

2.1 Defining k–CSP_d

Definition 1. *A k–CSP_d instance $\mathbb{L}$ consists of the following components:*
(1) A set $\mathbb{U}$ of variables
(2) The alphabet $\mathbb{A} = \{1, 2, \cdots, d\}$ from which the variables take on values
(3) A set $\mathbb{C}$ of constraints which operate on k variables each

We denote $n = |\mathbb{U}|$. We define a constraint as follows:

Definition 2. *A constraint $c \in \mathbb{C}$ consists of*
(1) An ordered set $V(c)$ of k variables $(u_1, u_2, \cdots, u_k)$
(2) A subset $S(c) \subset \mathbb{A}^k$ of satisfying assignments

Definition 3. *We refer to $|S(c)|$ as $\rho(c)$, or the density of c, and to the elements of $S(c)$ as $s_1, s_2, \cdots, s_{\rho(c)}$.*

We define an assignment of the variables as follows:

Definition 4. *An assignment of the variables is a mapping $x : \mathbb{U} \to \mathbb{A}$ which specifies the value assigned to each variable of the k–CSP_d.*

Finally, we use the following terminology to discuss constraint satisfaction:

Definition 5. *Consider any assignment x and any clause c with $V(c) = (u_1, u_2, \cdots, u_k)$. We say that x satisfies c if and only if $(x(u_1), x(u_2), \cdots, x(u_k)) \in S(c)$.*

Definition 6. *The value of an assignment x is the number of constraints that x satisfies. $OPT(\mathbb{L})$ is the maximum value of any assignment of the variables of $\mathbb{L}$.*

2.2 Reduction to the case of k–AND

Before we approximate the k–CSP_d problem, we simplify to the case where each clause is simply a k-AND clause. Since this is not a new idea, but was used in [15] and many previous works, we defer the exact method to the appendix. However, from here on, we consider only instances in which every clause is a k–AND clause.

2.3 The SDP

We approach the problem by using the SDP of Makarychev and Makarychev [15]. First, we need the following definitions.

Definition 7. *For any variable $u \in \mathbb{U}$ and clause $c \in \mathbb{C}$, we say that $(u, i) \in supp(c)$ if $u \in V(c)$ and i is the assignment to the variable u that satisfies c.*

Definition 8. *For any vector u, $||u||$ refers to the l_2 norm of u. Thus, $||u||^2$ refers to the squared l_2 norm of u, which we refer to as the magnitude of u.*

Definition 9. *For vectors u and v, $\langle u, v\rangle$ refers to the inner product of u and v.*

In the SDP, there is one vector z_c for each clause $c \in \mathbb{C}$. Also, for each variable $u \in \mathbb{U}$, there are d vectors $z_{u,1}, z_{u,2}, \cdots, z_{u,d}$, each of which represents one possible assignment of u.

Definition 10. *The canonical semidefinite program, usually referred to as simply the SDP, is given by*

$$\begin{aligned}
&\textit{Maximize} \sum_{c\in\mathbb{C}} ||z_c||^2 \textit{ such that}\\
&\langle z_c, z_{u,i}\rangle = ||z_c||^2 \textit{ for all } c \in \mathbb{C}, (u,i) \in \textit{supp}(c)\\
&\langle z_c, z_{u,i}\rangle = 0 \textit{ for all } c \in \mathbb{C}, u \in V(c), (u,i) \notin \textit{supp}(c)\\
&\langle z_{u,i}, z_{u,j}\rangle = 0 \textit{ for all } u \in \mathbb{U}, i \neq j\\
&\sum_{i=1}^{d} ||z_{u,i}||^2 \leq 1 \textit{ for all } u \in \mathbb{U}
\end{aligned}$$

Definition 11. *We denote by $SDP(\mathbb{L})$ the optimal value of this SDP for the k–CSP_d $\mathbb{L}$.*

Note that the constraints give us for any feasible solution that, for all $c \in \mathbb{C}$ and $(u,i) \in \textit{supp}(c)$,

$$\langle z_c, z_{u,i}\rangle = ||z_c||^2 \implies ||z_c|| \cdot ||z_{u,i}|| \geq ||z_c||^2 \implies ||z_{u,i}|| \geq ||z_c||.$$

Thus, if a clause vector is long, and thus contributes highly to the SDP value, then it also forces its satisfying assignments to be long. Since $||z_{u,i}||^2 \leq 1$ for all u, i, this also implies that for all $c \in \mathbb{C}$, $||z_c||^2 \leq 1$.

For a proof that this SDP is indeed a relaxation of the k–CSP_d problem, so that $SDP(\mathbb{L}) \geq OPT(\mathbb{L})$, see Section 2.1 of [15].

Now, we define several more pieces of useful terminology. First, we classify each clause c of the k–CSP_d in terms of the magnitude $||z_c||^2$ of its clause vector in the SDP.

Definition 12. *Let a clause c be short if $||z_c||^2 \leq \frac{64}{kd}$.*

Definition 13. *Let a clause c be long if $||z_c||^2 \geq \frac{2}{d}$.*

Definition 14. *Let a clause c be medium if it is neither short nor long.*

Definition 15. *Let a clause c be (α, β)–bounded if $\alpha \leq ||z_c||^2 \leq \beta$.*

Now, depending on how many clauses of the k–CSP_d fall into each of these categories, we will use different algorithms to achieve strong approximation ratio results. To discuss this, we define the following terminology.

Suppose that $\mathbb{L}$ is a k–CSP_d instance and x is a real number with $0 \leq x \leq 1$. Also, suppose that $T \subset \mathbb{C}$ specifies a subset of the clauses of $\mathbb{L}$.

Definition 16. *$\mathbb{L}$ is x-dense in clauses of type T if*

$$\sum_{c \in T} ||z_c||^2 \geq xSDP(\mathbb{L}).$$

Note that $\mathbb{L}$ is 1–dense in clauses of type $\mathbb{C}$.

Usually, we do not directly specify the subset T as such. For example, we say that $\mathbb{L}$ is x-dense in clauses which are long if

$$\sum_{c \text{ is } long} ||z_c||^2 \geq xSDP(\mathbb{L}).$$

Finally, before proceeding to present our algorithm, we prove the following lemma which we use throughout our analysis.

Lemma 1. *Suppose that a k–CSP_d instance $\mathbb{L}$ is r–dense in clauses of type T. Also, suppose that an algorithm R satisfies every clause $c \in T$ with a probability of at least $q||z_c||^2$. Then, R has an expected approximation ratio of at least rq.*

Proof. The expected value of the assignment output by R is at least

$$\sum_{c \in T} Pr(\text{R satisfies c})$$

By the hypotheses of the theorem, we have

$$\sum_{c \in T} Pr(\text{R satisfies c}) \geq \sum_{c \in T} q||z_c||^2 = q \sum_{c \in T} ||z_c||^2 \geq rqSDP(\mathbb{L}).$$

Thus, R achieve an expected approximation ratio of at least

$$rqSDP(\mathbb{L})/OPT(\mathbb{L}) \geq rqSDP(\mathbb{L})/SDP(\mathbb{L}) = rq.$$

□

3 The Algorithm

In this section, we present our k–CSP_d algorithm, R. Algorithm R is primarily a combination of three algorithms, A, B, and C, which each handle special cases of k–CSP_d. Algorithm A handles cases in which many of the clause vectors are very short; B handles those in which many of the clause vectors are very long; and C handles those in which many of the clause vectors are medium. In addition to these three algorithms, R also relies on two other algorithms. For the case of constant k, we rely on the greedy algorithm, D, described in Section 7. For the case of constant d, we rely on the algorithm of Charikar, Makarychev, and Makarychev [8]. Now, we describe the steps of algorithms A, B, and C.

3.1 Algorithm *A*:

Algorithm A outputs a random assignment.

1. For every $u \in \mathbb{U}$, determine $x(u)$ uniformly at random among $\{1, 2, \cdots, d\}$.
2. Output the assignment x.

3.2 Algorithm *B*:

Algorithm B is the main contribution of this paper. Step 1 calculates a value that is needed in the rest of the algorithm. Step 2 defines the set of ranges $[x_i, x_{i-1}]$ that we use to partition the long clauses. Step 3 finds a range which has enough clauses in it to give us the approximation ratio we need. Step 4 chooses variable assignments so as to maximize our chances of satisfying clauses in that range. Finally, step 5 outputs the result.

1. Calculate the maximum γ such that $\mathbb{L}$ is γ–dense in clauses which are long.
2. Let $b = 0$, $x_b = 1$. Until $x_b < \frac{2}{d}$:
   (a) Increment b by 1.
   (b) Calculate the sequence $\{r_i\}_{1 \leq i \leq b}$ defined by $r_i = \frac{k^{b+1-i}}{k-1}$.
   (c) Calculate the decreasing sequence $\{x_i\}_{1 \leq i \leq b}$ recursively so that $x_0 = 1$ and $x_i^k = k r_i d^{-k+1} x_{i-1}$ for $i > 0$.
3. Determine i with $0 < i \leq b$ such that $\mathbb{L}$ is $\frac{\gamma}{r_i}$–dense in clauses which are (x_i, x_{i-1})–bounded.
4. Create an assignment z. For each variable u:
   (a) Eliminate every assignment j such that $||z_{u,j}||^2 < x_i$.
   (b) Choose $z(u)$ uniformly at random from the remaining assignments if there are any. Otherwise, choose $z(u)$ arbitrarily.
5. Output the resulting assignment z.

3.3 Algorithm *C*:

See Section 6.

3.4 Algorithm *R*:

R simply runs the other algorithms and outputs the best result.

1. Run A, B, and C; record the outputted assignments x_a, x_b, and x_c.
2. Run the greedy algorithm D and record the outputted assignment x_d.
3. Run the algorithm of Charikar, Makarychev, and Makarychev [8], and record the outputted assignment x_e.
4. Output the assignment with the largest value among x_a, x_b, x_c, x_d, and x_e.

4 Proof of Approximation Ratio

Our main result is the following theorem:

Theorem 1. *R approximates any k–CSP_d instance with an expected approximation ratio of $\Omega(kd/d^k)$.*

4.1 Auxiliary Theorems

In order to prove this theorem, we prove the following theorems about A, B, and C. Theorem 2 says that A suffices when $\mathbb{L}$ is $\frac{1}{3}$–dense in short clauses. This is because short clauses provide an upper bound on the SDP value which gives the random assignment strategy just enough of an edge to provide the needed approximation ratio. Although this theorem is not novel, due to its brevity we give its full proof in this section.

Theorem 3 says that, if $\mathbb{L}$ is $\frac{1}{3}$–dense in long clauses, B suffices. Here, when the clause vectors are long, is the case that Makarychev and Makarychev could not handle for large d. Their strategy for this case is to simply apply the same algorithm that works well for certain medium clauses; however, this does not yield a high enough approximation ratio for long clauses and large d.

To remedy this problem, our algorithm B uses a spectrum of threshold lengths and finds a pair of adjacent thresholds such that many assignment vector lengths lie between the two. We then use the lower threshold as a cut-off for vector selection, and the upper threshold to provide a tighter upper bound on the SDP value. By carefully tuning the thresholds to be just close enough together to allow an effective algorithm for vectors between any pair of adjacent thresholds, but just far enough apart that they still span the entire range of possible lengths, we produce the first effective algorithm for this case. Since this algorithm is the primary contribution of this paper, we present the important parts of its proof in this section and try to provide ample motivation and explanation. However, for the sake of brevity, we postpone the rather technical proofs of Lemmas 2 and 3 to the appendix.

Finally, Theorem 4 says that, if $\mathbb{L}$ is $\frac{1}{3}$–dense in medium clauses, C suffices. Here, the intuition is that, among clauses that lie in this small intermediate length range, inner products faithfully represent the directions that clause vectors point in, so taking inner products with a random gaussian vector is a viable strategy.

4.2 Proof of Theorems

Theorem 2. *For any k–CSP_d instance which is $\frac{1}{3}$–dense in clauses which are short, A yields an expected approximation ratio of $\Omega(kd/d^k)$.*

Proof. Recall that A simply assigns each variable at random. Thus, A satisfies each clause c with a probability of exactly d^{-k}. For any clause c which is short, we have $||z_c||^2 \leq \frac{64}{kd}$, so that $d^{-k} \geq \frac{1}{64} kd/d^k ||z_c||^2$.

Thus, by Lemma 1, A has an expected approximation ratio of at least

$$\frac{1}{3}\frac{1}{64}kd/d^k = \frac{1}{192}kd/d^k$$

□

This brings us to the main result of this paper; namely, the proof that B handles instances with many long clause vectors:

Theorem 3. *For any $k \geq 18$, and any k–CSP_d instance which is $\frac{1}{3}$–dense in clauses that are long, B achieves an expected approximation ratio of $\Omega(kd/d^k)$.*

We require the following two lemmas, proved in Section 5.

Lemma 2. *For any $k \geq 18$, and some value of b polynomial in k and d, the intervals $[x_i, x_{i-1}]$ are all well defined, as $x_0 > x_1 > \cdots > x_b$, and $x_b < \frac{2}{d}$.*

Lemma 3. *For any b,*

$$\sum_{i=1}^{b} \frac{1}{r_i} \leq 1.$$

Proof of Theorem 3. First, by Lemma 2, we know that, for $k \geq 18$, step 2 of B terminates in polynomial time, since there is a polynomial value of b for which $x_b < \frac{2}{d}$. Also, we know that the intervals $[x_i, x_{i-1}]$ partition the interval $[\frac{2}{d}, 1]$, so that any clause c which is long must, for some i, be (x_i, x_{i-1})–bounded. For each i, let f_i be the maximum density such that $\mathbb{L}$ is f_i–dense in clauses that are (x_i, x_{i-1})–bounded.

Since $\mathbb{L}$ is γ–dense in clauses that are long, and every such clause is counted by some f_i, we know that

$$\sum_i f_i \geq \gamma.$$

Therefore, by Lemma 3,

$$\sum_i f_i \geq \gamma \sum_i \frac{1}{r_i} = \sum_i \frac{\gamma}{r_i}.$$

Thus, there must be some value of i such that $f_i \geq \frac{\gamma}{r_i}$, and $\mathbb{L}$ is $\frac{\gamma}{r_i}$–dense in clauses which are (x_i, x_{i-1})–bounded, so step 3 of algorithm B is always possible.

Now, consider any clause c which is (x_i, x_{i-1})–bounded, and some variable $u \in V(c)$. Since $||z_c||^2 \geq x_i$, we know that, for any $(u, j) \in supp(c)$, $||z_{u,j}||^2 \geq ||z_c||^2 \geq x_i$, so that j is among the values that B may choose for variable u in step 4. In addition, since

$$\sum_{k=1}^{d} ||z_{u,k}||^2 = 1,$$

there can be at most $\frac{1}{x_i}$ values k with $||z_{u,k}||^2 \geq x_i$, so that B chooses j with a probability of at least x_i. Thus, B satisfies c with a probability of at least x_i^k. By the definition of x_i, and the fact that $||z_c||^2 \leq x_{i-1}$,

$$Pr(B \text{ satisfies } c) \geq x_i^k = kr_i d^{-k+1} x_{i-1} \geq r_i kd/d^k ||z_c||^2$$

$\mathbb{L}$ is $\frac{\gamma}{r_i}$–dense in such clauses, and, since $\mathbb{L}$ is $\frac{1}{3}$–dense in long clauses, we know that $\gamma \geq \frac{1}{3}$. Lemma 1 then tells us that B has an approximation ratio of at least

$$\frac{1}{3r_i} r_i kd/d^k = \frac{1}{3} kd/d^k.$$

This completes the proof of Theorem 3. □

Now, we have the following theorem for k–CSP_d instances with many medium length vectors.

Theorem 4. *For any $k \geq 44$, $d \geq 72$, and any k–CSP_d which is $\frac{1}{3}$–dense in clauses which are medium, C achieves an expected approximation ratio of $\Omega(kd/d^k)$.*

Proof. See Section 6. □

Given Theorems 2, 3, and 4, it is straightforward to prove our main theorem. First, recall the statement of the theorem:

Theorem 1. *R approximates any k–CSP_d instance with an expected approximation ratio of $\Omega(kd/d^k)$.*

Proof. If $k \leq 44$, the greedy assignment x_d will have an approximation ratio of $\Omega(d/d^k) = \Omega(kd/d^k)$. If $d \leq 72$, the assignment x_e will have an approximation ratio of $\Omega(k/d^k) = \Omega(kd/d^k)$.

For $k \geq 44$, $d \geq 72$, we consider three cases. First, it could be that the k–CSP_d instance $\mathbb{L}$ is $\frac{1}{3}$–dense in clauses which are short. In this case, Theorem 2 guarantees an approximation ratio of at least $\Omega(kd/d^k)$. Otherwise, $\mathbb{L}$ may be $\frac{1}{3}$–dense in clauses which are long. In this case, Theorem 3 guarantees an approximation ratio of at least $\Omega(kd/d^k)$.

Now, suppose that $\mathbb{L}$ is neither $\frac{1}{3}$–dense in clauses that are short, nor $\frac{1}{3}$–dense in clauses that are long. We know that every clause which is neither short nor long is medium, and we have $1 - \frac{1}{3} - \frac{1}{3} = \frac{1}{3}$. Thus, $\mathbb{L}$ must be $\frac{1}{3}$–dense in clauses which are medium. Theorem 4 then guarantees an approximation ratio of at least $\Omega(kd/d^k)$, and we are done. □

References

1. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.
2. S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.
3. S. Arora and M. Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.
4. P. Austrin and E. Mossel. Approximation resistant predicates from pairwise independence. *computational complexity*, 18(2):249–271, 2009.
5. L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 21–32, 1991.
6. L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.
7. S. O. Chan. Approximation resistance from pairwise independent subgroups. In *Proc. 45th ACM Symp. on Theory of Computing*, pages 447–456, 2013.
8. M. Charikar, K. Makarychev, and Y. Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Trans. Algorithms*, 5(3):32:1–32:14, 2009.
9. I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra. PCP characterizations of NP: Toward a polynomially-small error-probability. *Computational Complexity*, 20(3):413–504, 2011.
10. L. Engebretsen. The nonapproximability of non-boolean predicates. *SIAM J. Discrete Math.*, 18(1):114–129, 2004.
11. L. Engebretsen and J. Holmerin. More efficient queries in PCPs for NP and improved approximation hardness of maximum CSP. *Random Structures and Algorithms*, 33(4):497–514, 2008.
12. U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.
13. V. Guruswami and P. Raghavendra. Constraint satisfaction over a non-boolean domain: Approximation algorithms and unique-games hardness. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 5171 of *Lecture Notes in Computer Science*, pages 77–90. 2008.
14. G. Hast. Approximating MAX kCSP - outperforming a random assignment with almost a linear factor. In *In Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, pages 956–968, 2005.
15. K. Makarychev and Y. Makarychev. Approximation algorithm for non-boolean MAX k-CSP. *Theory of Computing*, 10:341–358, 2014.
16. D. Moshkovitz and R. Raz. Two query PCP with sub-constant error. *Journal of the ACM*, 57(5), 2010.
17. R. Raz and S. Safra. A sub-constant error-probability low-degree test and a sub-constant error-probability PCP characterization of NP. In *Proc. 29th ACM Symp. on Theory of Computing*, pages 475–484, 1997.
18. A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proc. 32nd ACM Symp. on Theory of Computing*, pages 191–199, 2000.

19. A. Samorodnitsky and L. Trevisan. Gowers uniformity, influence of variables, and PCPs. In *Proc. 38th ACM Symp. on Theory of Computing*, pages 11–20, 2006.
20. L. Trevisan. Parallel approximation algorithms by positive linear programming. *Algorithmica*, 21(1):72–88, 1998.
21. Z. Šidák. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62(318):626–633, 1967.

Appendix

5 Proofs of Lemmas Required by Theorem 3

Lemma 2. *For any $k \geq 18$ and some value of b polynomial in k and d, the intervals $[x_i, x_{i-1}]$ are all well defined, as $x_0 > x_1 > \cdots > x_b$, and $x_b < \frac{2}{d}$.*

Proof. We first find an explicit formula for x_i. To express this somewhat cleanly, we define the following two series:

$$g_i = \sum_{j=1}^{i} k^{-j}$$

$$h_i = \sum_{j=1}^{i} jk^{-j}$$

For convenience, we set $g_0 = h_0 = 0$.

Claim. $x_i = (k-1)^{-g_i} k^{g_i(b-i+1)+h_i} d^{-1+k^{-i}}$

Proof. We proceed by induction. For $i = 0$, our formula says that

$$\begin{aligned} x_0 &= (k-1)^{-g_i} k^{g_i(b-i+1)+h_i} d^{-1+k^{-i}} \\ &= (k-1)^0 k^0 d^0 \\ &= 1, \end{aligned}$$

which completes the base case.

Now, we assume that the proposed formula holds for x_i, and prove that it holds for x_{i+1}. Note first the following helpful identities, which hold for all $i \geq 0$:

$$g_{i+1} = \frac{1+g_i}{k}, \text{ and } h_{i+1} = \frac{1+h_i+g_i}{k}.$$

Then we have

$$\begin{aligned} x_{i+1} &= (r_{i+1} k x_i)^{\frac{1}{k}} d^{-1+\frac{1}{k}} \\ &= \left(\frac{k^{b-i}}{k-1} k (k-1)^{-g_i} k^{g_i(b-i+1)+h_i} d^{-1+k^{-i}} \right)^{1/k} d^{-1+\frac{1}{k}} \\ &= \left((k-1)^{-1-g_i} k^{(b-i+1)+h_i+g_i(b-i+1)} \right)^{1/k} d^{-1+k^{-(i+1)}} \\ &= (k-1)^{-(1+g_i)/k} k^{(1+h_i+g_i)/k+(b-i)(1+g_i)/k} d^{-1+k^{-(i+1)}} \\ &= (k-1)^{-g_{i+1}} k^{h_{i+1}+(b-(i+1)+1)g_{i+1}} d^{-1+k^{-(i+1)}} \end{aligned}$$

This completes the inductive step. □

We now bound the expression for x_b based on bounds on g_i and h_i. Note that $g_i < \frac{1}{k-1}$ for all i, and $h_i < \frac{k}{(k-1)^2}$ for all i. Thus,

$$\begin{aligned} x_b &= d^{-1+k^{-b}}(k-1)^{-g_b}k^{h_b+g_b} \\ &< \frac{1}{d}\left(d^{k^{-b}}\right)\left(k^{(2k-1)/(k-1)^2}\right) \end{aligned}$$

As k grows, the third term approaches 1. In particular, for $k \geq 18$, it is easy to check that $k^{(2k-1)/(k-1)^2} \leq \sqrt{2}$. Also, given fixed values of k and d, we can find a value of b which is on the order of $\frac{\log\log d}{\log k}$ (importantly, polynomial in d, k) such that $d^{k^{-b}} < \sqrt{2}$. This gives us, for all $k \geq 18$, that $x_b < \frac{2}{d}$, as desired.

The last bit of work is to show that the x_i's monotonically decrease, so that the intervals we describe make sense. To see this, recall that, by definition,

$$x_i = (\frac{r_i k}{\delta} d^{-k+1} x_{i-1})^{1/k}$$

Clearly, x_i is a monotonically increasing function of x_{i-1}.

Thus, if $x_1 > x_0$, then $x_i > x_{i-1}$ for all i, and similarly, if $x_1 < x_0$, then $x_i < x_{i-1}$ for all i. Since we now know that, for the relevant values, $x_b < \frac{2}{d} \leq 1 = x_0$, it must be that $x_i < x_{i-1}$ for all i, so we are done.

□

Remark 1. In the case $k = \Omega(\log d)$, we get $b = 1$, and our algorithm degenerates into one that differs only trivially from that of Makarychev and Makarychev.

To see this, note that, in the proof of Lemma 2, we found the value of b to be $O\left(\frac{\log\log d}{\log k}\right)$. Thus, if $k = \Omega(\log d)$, for an appropriate constant factor, we have $b = 1$, in which case our spectrum of possible thresholds degenerates into a single-threshold approach, which is not substantially different than the approach of Makarychev and Makarychev.

Lemma 3. *For any b,*

$$\sum_{i=1}^{b} \frac{1}{r_i} \leq 1.$$

Proof. Simple algebra yields the lemma.

$$\sum_{i=1}^{b} \frac{1}{r_i} = \sum_{i=1}^{b} \frac{k-1}{k^{b+1-i}} = (k-1)\sum_{i=1}^{b} k^{-i} \leq (k-1)\sum_{i=1}^{\infty} k^{-i} = 1$$

□

6 Algorithm C and the Proof of Theorem 4

First, we introduce several definitions which will be useful in this section. Many of these definitions are either taken directly from Makarychev and Makarychev, or are extensions or generalizations of their definitions.

Definition 17. *Let ξ be a Gaussian random variable with mean 0 and standard deviation 1. Then, $\Phi(t) = Pr(|\xi| \leq t)$, and $\bar{\Phi}(t) = 1 - \Phi(t)$.*

We utilize the following inequality, which is proved and similarly used in [15]:

Theorem 5. *$\bar{\Phi}(\beta t) \leq \bar{\Phi}(t)^{\beta^2}$ for all $t \geq 0$, $\beta \in (0, 1]$.*

We also utilize a theorem of Šidák, proved in [21] and utilized similarly in [15]:

Theorem 6. *Let $\xi_1, \xi_2, \cdots, \xi_r$ be Gaussian random variables with mean zero and arbitrary covariance matrix. Then for any positive $t_1, t_2, \cdots, t_r$,*

$$Pr(|\xi_1| \leq t_1, |\xi_2| \leq t_2, \cdots, |\xi_r| \leq t_r) \geq \prod_{i=1}^{r} Pr(|\xi_i| \leq t_i)$$

Now, we define, for each variable $u \in \mathbb{U}$, a partition of the alphabet $\mathbb{A}$. $S(u)$ contains the $\frac{d}{2}$ shortest-vectored assignments of the variable u, and $L(u)$ contains the $\frac{d}{2}$ longest-vectored assignments. Note that this is completely different from our previous hierarchy of short, medium, and long vectors, which applied uniformly to clause vectors rather than separately to each variable's set of assignment vectors.

Definition 18. *For every $u \in \mathbb{U}$, define the set of short-vectored assignments*
$S(u) = \{i \in \mathbb{A} \mid ||z_{u,i}|| \geq ||z_{u,j}|| \text{ for at most } \frac{d}{2} \text{ values of } j \in \mathbb{A}\}$, *and define the set of long-vectored assignments*
$L(u) = \mathbb{A}/S(u)$

Note that $|S(u)| = \lfloor \frac{d}{2} \rfloor$ and $|L(u)| = \lceil \frac{d}{2} \rceil$. Also note that, if $i \in S(u)$, it must be that $||z_{u,i}||^2 \leq \frac{2}{d}$, since

$$1 \geq \sum_{j \in L(u)} ||z_{u,j}||^2 \geq \sum_{j \in L(u)} ||z_{u,i}||^2 = \lceil \frac{d}{2} \rceil ||z_{u,i}||^2 \geq \left(\frac{d}{2}\right) ||z_{u,i}||^2.$$

We now return to considering the assignments which satisfy a single clause, rather than those corresponding to a single variable, and note that there can be arbitrarily many short or long assignments satisfying that clause. To quantify this, we define a measure called the *lightness* of a clause to specify how many of the vectors $z_{u,i}$, for $(u, i) \in supp(c)$, are short. This notion runs parallel to our original hierarchy of short, medium, and long clause vectors as an alternative way of classifying clauses based on the SDP.

Definition 19. *Define the lightness $l(c)$ of a clause c by* $l(c) = |\{i|(u,i) \in supp(c) \text{ and } i \in S(u)\}|$.

Definition 20. *Let a light clause c be one such that $l(c) \geq \frac{k}{4}$.*
Let a heavy clause be one which is not light.

Note that the criterion for a clause to be light is, in a sense, weak, since we would expect on average each clause to have a lightness $l(c) \approx \frac{k}{2}$.

Now, recall the statement of the theorem:

Theorem 4. *For any $k \geq 44$, $d \geq 72$, and any k–CSP_d which is $\frac{1}{3}$–dense in clauses which are medium, C achieves an expected approximation ratio of $\Omega(kd/d^k)$.*

We now present the full details of algorithm C. This algorithm is essentially taken from the paper of Markarychev and Makarychev [15], but since we require a restructuring of both the algorithm and the analysis, we reproduce both here. C and is itself the combination of two algorithms, $C1$ and $C2$, which each output an assignment.

6.1 Algorithm $C1$:

1. Let y be an assignment such that, for every $u \in \mathbb{U}$, $y(u)$ is with probability $\frac{|S(u)|}{2d}$ chosen at random from $S(u)$, and with the remaining probability chosen at random from $L(u)$.

6.2 Algorithm $C2$:

1. Choose a random Gaussian vector g so that every component of g is distributed as an independent Gaussian variable with mean 0 and variance 1.
2. Let z be a uniformly random assignment of every variable $u \in \mathbb{U}$.
3. Let x be an assignment such that, for every $u \in \mathbb{U}$,
   $x(u) = z(u)$ if $z(u) \in L(u)$, and
   $x(u) = \arg\max_{i \in S(u)} |\langle z_{u,i}, g \rangle|$ otherwise.

6.3 Algorithm C:

1. Run $C1$ and $C2$; record the outputted assignments x_1 and x_2.
2. Output the assignment with the larger value among x_1 and x_2.

Intuitively, $C1$ chooses assignments from a simple random distribution where values in $L(u)$ are chosen more often and values in $S(u)$ are chosen less often than in the uniform distribution. This means that clauses for which many satisfying assignments lie in $L(u)$ are satisfied with a high probability by $C1$.

On the other hand, $C2$ is tuned to do well on clauses which have many satisfying assignments among $S(u)$. First, it uses the partial assignment z to eliminate all of the assignments in $L(u)$ for every variable u. Then, among the clauses which have not yet been violated, it uses projections onto a random Gaussian vector to select its assignments, which performs well because none of the remaining assignment vectors are too long. C then chooses the better of these two strategies.

6.4 Proof of Theorem 4

Proof. We divide k–CSP_d instances into the following two cases:

Case 1. $\mathbb{L}$ is $\frac{1}{6}$–dense in clauses which are both medium and heavy.

Case 2. $\mathbb{L}$ is $\frac{1}{6}$–dense in clauses which are both medium and light.

We show that, in Case 1, $C1$ achieves the desired approximation ratio, and in Case 2, $C2$ achieves the desired approximation ratio.

Lemma 4. *For any k–CSP_d in Case 1 which satisfies the conditions of Theorem 4, $C1$ achieves an expected approximation ratio of at least $\Omega(kd/d^k)$.*

Proof. Consider a heavy, medium clause c, and some $u \in V(c)$, and suppose that $(u, i) \in supp(c)$. Then, if $i \in S(u)$, $C1$ chooses i with a probability of

$$\frac{|S(u)|}{2d}\frac{1}{|S(u)|} = \frac{1}{2d}.$$

If $i \in L(u)$, $C1$ chooses i with a probability of

$$\frac{2d - |S(u)|}{2d}\frac{1}{|L(u)|} \geq \frac{3|S(u)|}{2d|L(u)|} \approx \frac{3}{2d}$$

In particular, for $d \geq 72$, it is easy to check that this value is at least $\frac{\sqrt{2}}{d}$.

Thus, since there are $l(c)$ assignments $(u, i) \in supp(c)$ such that $i \in S(u)$, we have

$$Pr(c \text{ is satisfied}) \geq \left(\frac{1}{2}d^{-1}\right)^{l(c)} \left(\sqrt{2}d^{-1}\right)^{k-l(c)} = d^{-k}\left(\frac{1}{2}\right)^{l(c)} \left(\sqrt{2}\right)^{k-l(c)}$$

This is monotonically decreasing in $l(c)$, so we can lower bound it for δ–heavy clauses c by rounding $l(c)$ up to $\frac{k}{4}$ to get

$$Pr(c \text{ is satisfied}) \geq d^{-k}\left(\frac{1}{2}\right)^{\frac{k}{4}} \left(\sqrt{2}\right)^{\frac{3k}{4}} \geq d^{-k}2^{k/8}$$

It is easy to check that, for $k \geq 44$, we have

$$2^{k/8} \geq k.$$

Thus, for all $k \geq 44$,

$$Pr(c \text{ is satisfied}) \geq k/d^k.$$

Since $||z_c||^2 \leq \frac{2}{d}$ for any medium clause c, we can write this as

$$Pr(c \text{ is satisfied}) \geq \frac{1}{2} kd/d^k ||z_c||^2.$$

Thus, since $\mathbb{L}$ is $\frac{1}{6}$–dense in clauses which are heavy and medium, Lemma 1 tells us that $C1$ achieves an expected approximation ratio of at least $\frac{1}{12} kd/d^k$. □

Lemma 5. *For any k–CSP_d in Case 2 which satisfies the conditions of Theorem 4, $C2$ achieves an expected approximation ratio of at least $\Omega(kd/d^k)$.*

Proof. First, recall that the medium clause vectors are those which satisfy

$$\frac{64}{kd} < ||z_c||^2 < \frac{2}{d}$$

Also, note that $C2$ only relies on the relative values of dot products of SDP vectors with g, so that it is invariant under scaling all of our SDP vectors by any constant. Thus, we may scale every vector by a factor of $\frac{kd}{64}$, so that medium clauses now satisfy

$$1 \leq ||z_c||^2 \leq \frac{k}{32}.$$

In analyzing $C2$, we imagine that it is carried out in two steps, so that first, we use the random assignment z to create a partial assignment of the variables, and then we use the Gaussian projection to complete the assignment. In the terminology of Makarychev and Makarychev, the partial assignment z allows us to transform the general SDP into a *uniform* SDP in which every variable vector $z_{u,i}$ is also bounded by $||z_{u,i}||^2 \leq \frac{k}{32}$. Again, using the terminology of Makarychev and Makarychev, we say that c *survives* z if every variable u constrained by c is either constrained by c to be among $S(u)$, and left unassigned by z, or is constrained by c to be among $L(u)$, and is assigned correctly by z. Thus the surviving clauses are exactly those which can still be satisfied by the Gaussian assignment given the choices of the random assignment. For a given clause c, $l(c)$ then gives us the size of any clause c which survives the partial assignment z.

Now, consider any light clause c, and let $s = l(c)$ for convenience. Recall that, since c is light, $s \geq \frac{k}{4}$. For any $u \in V(c)$ with $(u, i) \in supp(c)$, z has a probability of at least $\frac{1}{2}$ of leaving u unassigned if $i \in S(u)$, and a probability of $\frac{1}{d}$ of assigning it correctly if $i \in L(u)$. Thus, c survives with a probability of at least

$$\left(\frac{1}{d}\right)^{k-s} \left(\frac{1}{2}\right)^{s}.$$

Now, consider a survived light clause c in the new uniform SDP. Assume without loss of generality that for every $u \in V(c)$, $(u, 1) \in supp(c)$. We define the following vectors and values in order to allow us to analyze the Gaussian projection strategy.

First, we define a vector which represents the component of $z_{u,1}$ which is perpendicular to z_c.

Definition 21. *Let* $z^{\perp}_{u,1} = z_{u,1} - z_c$.

We can derive from the SDP constraints that $\langle z^{\perp}_{u,1}, z_c \rangle = 0$, as desired. Now, we define three values, each given by an inner product, will help us bound the probability that clause c is satisfied by the strategy that uses projections onto g.

Definition 22. *Let* $\gamma_{u,1} = \langle z^{\perp}_{u,1}, g \rangle$, *and for* $i \geq 2$, *let* $\gamma_{u,i} = \langle z_{u,i}, g \rangle$.

Definition 23. *Let* $\gamma_c = \langle g, z_c \rangle$.

Note that, since $z_c \perp z^{\perp}_{u,1}$, and $z_c \perp z_{u,i}$ for $i \neq 1$, γ_c is independent from every $\gamma_{u,i}$ variable.

Finally, the following definitions define other values we will require in our next proof.

Definition 24. *Let* $M = \bar{\Phi}^{-1}(d^{-s/2})$, *so that* $\bar{\Phi}(M) = d^{-s/2}$.

Then we have

$$Pr\left(x \text{ satisfies } c\right) = Pr\left(\arg\max_i |\langle z_{u,i}, g\rangle| = 1 \ \forall \ u \in V(c)\right)$$

Before we delve into the tedious calculations, the intuition here is that, because of the constraints on the SDP vectors, vector $z_{u,1}$ is guaranteed to have a component in the direction of z_c, while for every $i \neq 1$, $z_{u,i}$ is forbidden from having any such component. Thus, if g points roughly in the direction of z_c, we will have a good chance of picking $z_{u,1}$ for every $u \in V(c)$, which leads to an overall chance of satisfying c which is much higher than the d^{-k} chance given by a random selection.

To show this, we begin by breaking down the maximum into one inequality for each non-maximal value, and writing it in terms of our defined γ values:

$$\begin{aligned} Pr\left(x \text{ satisfies } c\right) &= Pr\left(|\langle g, z_{u,1}\rangle| > |\langle g, z_{u,i}\rangle| \ \forall \ u \in V(c), i \geq 2\right) \\ &= Pr\left(|\gamma_{u,1} + \gamma_c| > |\gamma_{u,i}| \ \forall \ u \in V(c), i \geq 2\right) \end{aligned}$$

Now, this expression is unwieldy because it depends on the magnitude of the sum of two Gaussian variables. To fix this, we introduce some slack into our bound by noting that

$$(|\gamma_c| > M) \text{ and } (|\gamma_{u,1}| < M/2)) \text{ and } (|\gamma_{u,i}| \leq M/2) \implies |\gamma_{u,1} + \gamma_c| > |\gamma_{u,i}|.$$

Using this, and the fact that γ_c is independent from every $\gamma_{u,i}$, we get

$$
\begin{aligned}
&Pr\left(x \text{ satisfies } c\right) \\
&\geq Pr\left(\left(|\gamma_c| > M\right) \text{ and } \left(|\gamma_{u,1}| \leq M/2\right)\right) \text{ and } \left(|\gamma_{u,i}| \leq M/2; \forall u \in V(c), i \geq 2\right) \\
&= Pr\left(|\gamma_c| > M\right) Pr\left(\left(|\gamma_{u,1}| \leq M/2\right)\right) \text{ and } \left(|\gamma_{u,i}| \leq M/2 \; \forall u \in V(c), i \geq 2\right),
\end{aligned}
$$

where the last step uses the independence of γ_c.

Then, by Theorem 6, we can turn this conjunction into a product:

$$
Pr\left(x \text{ satisfies } c\right) \geq Pr\left(|\gamma_c| > M\right) \prod_{u \in V(c)} \left(Pr\left(|\gamma_{u,1}| \leq M/2\right) \prod_{i \geq 2} Pr\left(|\gamma_{u,i}| \leq M/2\right) \right)
$$

To bound these terms, we bound the variances of the γ's. Recall that, for any $i \in S(u)$, we have $||z_{u,i}||^2 \leq \frac{2}{d}$ in the original SDP, and thus $||z_{u,i}||^2 \leq \frac{k}{32}$ in the scaled SDP. We then have

$$
\begin{aligned}
&Var[\gamma_c] = ||z_c||^2 \geq 1, \text{ and} \\
&Var[\gamma_{u,i}] = ||z_{u,i}||^2 \leq \frac{k}{32}, \text{ for } i \geq 2.
\end{aligned}
$$

Now, to bound $Var[\gamma_{u,1}]$, we use the fact that $\langle z_{u,1}, z_c \rangle = ||z_c||^2$ by the constraints of the SDP. Thus,

$$
Var[\gamma_{u,1}] = ||z_{u,1}^{\perp}||^2 = \langle z_{u,1} - z_c, z_{u,1} - z_c \rangle = ||z_{u,1}||^2 - 2||z_c||^2 + ||z_c||^2 \leq ||z_{u,1}||^2 \leq \frac{k}{32}
$$

In the following bound, we make use of Theorem 5, which the reader will recall states that $\bar{\Phi}(\beta t) \leq \bar{\Phi}(t)^{\beta^2}$ for all $t \geq 0$ and $\beta \in (0, 1]$. We bound the first term as follows:

$$
\begin{aligned}
Pr\left(|\gamma_{u,1}| < M/2\right) &= \Phi\left(M/(2\sqrt{Var[\gamma_{u,1}]})\right) \\
&\geq \Phi\left(M\sqrt{\frac{8}{k}}\right) \\
&\geq 1 - \bar{\Phi}(M\sqrt{\frac{8}{k}}) \\
&\geq 1 - \bar{\Phi}(M)^{\frac{8}{k}} \\
&= 1 - d^{-4s/k} \\
&\geq 1 - d^{-1}
\end{aligned}
$$

Note that, for $k \geq 8$, we have $\sqrt{\frac{8}{k}} \leq 1$, as required by Theorem 5. For the second term, we have exactly the same bound, since the threshold and variance are the same. Thus

$$Pr\left(|\gamma_{u,i}| < M/2\right) \geq 1 - d^{-1}.$$

Finally, we have

$$Pr\left(|\gamma_c| > M\right) = \bar{\Phi}(M/\sqrt{Var[\gamma_c]}) \geq \bar{\Phi}(M) = d^{-s/2}$$

Combining these results, we have

$$\begin{aligned} Pr\left(x \text{ satisfies } c\right) &\geq \left(\prod_{u \in V(c)} Pr\left(|\gamma_{u,1}| \leq M/2\right) \prod_{i \geq 2} Pr\left(|\gamma_{u,i}| \leq M/2\right)\right) Pr\left(|\gamma_c| > M\right) \\ &= Pr\left(|\gamma_{u,1}| \leq M/2\right)^s Pr\left(|\gamma_{u,i}| \leq M/2\right)^{s(d-1)} Pr\left(|\gamma_c| > M\right) \\ &\geq \left(1 - d^{-1}\right)^s \left(1 - d^{-1}\right)^{s(d-1)} d^{-s/2} \geq \left(\left(1 - d^{-1}\right)^d\right)^s d^{-s/2} \end{aligned}$$

For $d \geq 72$, it is easy to check that $\left(1 - \frac{1}{d}\right)^d \geq \frac{1}{3}$, so that

$$Pr\left(x \text{ satisfies } c\right) \geq 3^{-s} d^{-s/2} \geq (9d)^{-s/2}$$

Combining this result for a survived clause c with the probability that a clause survives, we see that

$$Pr(c \text{ is satisfied }) \geq (9d)^{-s/2} d^{-(k-s)} \left(\frac{1}{2}\right)^s \geq d^{-k} \left(\frac{d}{36}\right)^{s/2} \geq d^{-k} \left(\frac{d}{36}\right)^{k/8}$$

For $d \geq 72$, we have $\frac{d}{36} \geq 2$, so that

$$Pr(c \text{ is satisfied}) \geq d^{-k} 2^{k/8}.$$

Then, for $k \geq 44$, we have $2^{k/8} \geq k$, so that, finally,

$$Pr(c \text{ is satisfied}) \geq k/d^k.$$

Since every medium clause c has $||z_c||^2 \leq \frac{2}{d}$, we satisfy every clause which is both medium and light with a probability of at least

$$\frac{1}{2} kd/d^k ||z_c||^2,$$

and, by Lemma 1, we can achieve an approximation ratio of $\frac{1}{12} kd/d^k$.

□

Given these two lemmas, the proof of the theorem is straightforward. Since $\mathbb{L}$ is $\frac{1}{3}$–dense in clauses which are medium, and every clause is either heavy or light, $\mathbb{L}$ must be in either Case 1 or Case 2. In Case 1, Lemma 4 proves the theorem, and in Case 2, Lemma 5 proves the theorem. □

7 The Greedy Algorithm

For small k, we use a simple algorithm which is a combination of random and greedy assignments.

7.1 Algorithm D:

1. For every $u \in \mathbb{U}$, with probability $\left(1 - \frac{1}{k}\right)$, assign $x(u)$ at random.
2. Discard every clause which has been violated, and every clause which does not have exactly one constrained variable unassigned.
3. Solve the remaining trivial k–CSP_d instance optimally by assigning each variable to the value which satisfies the most of the surviving clauses.

Theorem 7. *Algorithm D achieves an approximation ratio of* $\Omega(d/d^k)$.

Proof. Consider any k–CSP_d instance $\mathbb{L}$ with an optimal assignment x^* of value OPT. We first show that the value of OPT can drop by at most a $O(d/d^k)$ factor in the initial assignment and discarding steps of the algorithm. To see this, consider any clause c satisfied by x^*. The probability that c survives the discarding step of the algorithm is the probability that $(k-1)$ of its variables are assigned correctly, and the final one is not assigned. The probability that exactly $(k-1)$ of the variables have been assigned is given by

$$k\left(\frac{1}{k}\right)\left(1-\frac{1}{k}\right)^{k-1} = \Omega(1).$$

Furthermore, given that this occurs, the probability that the $(k-1)$ assigned variables were assigned correctly is $d^{-k+1} = d/d^k$. Thus, the probability that c survives is at least $\Omega(d/d^k)$. Now, if we assign every remaining variable as x^* would have assigned it, we will obtain an assignment that satisfies every such surviving clause c, or namely $\Omega(d/d^k \cdot OPT)$ clauses in expectation. Thus, the optimal value of the remaining k–CSP_d must be at least this high. Furthermore, since every clause that remains constrains only one variable, step 3 of D trivially finds the optimal solution. Therefore, D finds an assignment of value $\Omega(d/d^k \cdot OPT)$, and obtains an approximation ratio of $\Omega(d/d^k)$. □

8 Reduction to the Case of k–AND

Consider a general k–CSP_d instance $\mathbb{L}$ and some clause $c \in \mathbb{C}$. We define a set of $\rho(c)$ new clauses $N(c) = \{c_1, c_2, \cdots, c_{\rho(c)}\}$ with $V(c_i) = V(c)$ and $S(c_i) = \{s_i\}$. We then construct a new k–CSP_d instance $\mathbb{L}'$ with the same variables and alphabet as $\mathbb{L}$. However, the clauses of $\mathbb{L}'$ are

$$\mathbb{C} = \bigcup_{c \in \mathbb{C}} N(c).$$

For any assignment x and clause c in $\mathbb{L}$, x satisfies c iff x satisfies exactly one clause in $N(c)$, and x does not satisfy c iff x satisfies no clause in $N(c)$. Thus every assignment has the same value with respect to $\mathbb{L}'$ and $\mathbb{L}$, so the two problems are equivalent.