The Complexity of Making Unique Choices: Approximating 1-in-k SAT

Venkatesan Guruswami*
Dept. of Computer Science & Engineering
University of Washington
Seattle, WA 98195.

Luca Trevisan
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720.

Abstract

We study the approximability of 1-in-kSAT, the variant of Max kSAT where a clause is deemed satisfied when precisely one of its literals is satisfied. We also investigate different special cases of the problem, including those obtained by restricting the literals to be unnegated and/or all clauses to have size exactly k. Our results show that the 1-in-kSAT problem exhibits some rather peculiar phenomena in the realm of constraint satisfaction problems. Specifically, the problem becomes *substantially* easier to approximate with perfect completeness as well as when negations of literals are not allowed.

1 Introduction

Boolean constraint satisfaction problems (CSP) arise in a variety of contexts and their study has generated a lot of algorithmic and complexity-theoretic research. An instance of a Boolean CSP is given by a set of variables and a collection of Boolean constraints, each on a certain subset of variables, and the objective is to find an assignment to the variables that satisfies as many constraints as possible. The most fundamental problem in this framework is of course Max SAT where the constraints are disjunctions of a subset of literals (i.e., variables and their negations), and thus are satisfied if *at least* one of the literals in the subset are set to 1. When the constraints are disjunctions of at most k literals, we get the Max kSAT problem.

In this work we consider constraint satisfaction problems where each constraint requires that *exactly* one literal (from the subset of literals constrained by it) is set to 1. This is a natural variant of SAT which is also NP-hard. We study the version of this problem when all constraints contain at most k literals (for constant $k \geq 3$). We call this problem 1-in-kSAT. For $k = 3$, this problem has often found use as a more convenient starting point compared to 3SAT for NP-completeness reductions. For each $k \geq 3$, determining if all constraints can be satisfied is NP-hard [9]. We study the approximability of this problem and some of its variants.

In addition to being a natural satisfiability problem that merits study for its own sake, the underlying problem of making unique choices from certain specified subsets is salient to a natural pricing problem. We describe this connection at the end of the introduction.

We now formally define the problems we consider. Let $\{x_1, x_2, \ldots, x_n\}$ be a set of Boolean variables. For a set $S \subseteq \{x_i, \overline{x}_i \mid 1 \leq i \leq n\}$ of literals, the constraint ONE(S) is satisfied by an assignment to $\{x_1, \ldots, x_n\}$ if exactly one of the literals in S is set to 1 and the rest are set to 0.

*Supported in part by NSF Career Award CCF-0343672.

Definition 1 (1-in-kSAT **and** 1-in-EkSAT) *Let $k \geq 2$. An instance of* 1-in-kSAT *consists of a set $\{x_1, x_2, \ldots, x_n\}$ of Boolean variables, and a collection of constraints* ONE(S_i)*, $1 \leq i \leq m$ for some subsets $S_1, S_2, \ldots, S_m$ of $\{x_i, \bar{x}_i \mid 1 \leq i \leq n\}$, where $|S_i| \leq k$ for each $i = 1, 2, \ldots, m$. When each S_i has size* exactly *k, we get an instance of Exact 1-in-k Satisfiability, denoted* 1-in-EkSAT*.*

Definition 2 (1-in-kHS **and** 1-in-EkHS) *Let $k \geq 2$. An instance of* 1-in-kHS *consists of a set $\{x_1, x_2, \ldots, x_n\}$ of Boolean variables, and a collection of constraints* ONE(S_i)*, $1 \leq i \leq m$ for some subsets $S_1, S_2, \ldots, S_m$ of $\{x_i \mid 1 \leq i \leq n\}$, where $|S_i| \leq k$ for each $i = 1, 2, \ldots, m$. When each S_i has size* exactly *k, we get an instance of Exact 1-in-k Hitting Set, denoted* 1-in-EkSAT*.*

Note that 1-in-kHS (resp. 1-in-EkHS) is a special case of 1-in-kSAT (resp. 1-in-EkSAT) where no negations are allowed. Also, 1-in-kHS is simply the variant of the Hitting Set problem where we are given a family of sets each of size at most k from a universe, and the goal is to pick a subset of the universe which intersects a maximum number of sets from the input family in *exactly* one element. The 1-in-EkHS problem corresponds to the case when each set in the family has exactly k elements. For every $k \geq 3$, 1-in-EkHS is NP-hard [9].

Clearly, the following are both orderings of the problems in decreasing order of their generality: (i) 1-in-kSAT, 1-in-EkSAT, 1-in-EkHS; and (ii) 1-in-kSAT, 1-in-kHS, 1-in-EkHS.

The 1-in-3SAT problem was considered in Schaefer's work on complexity of satisfiability problems [9]. An inapproximability factor of $6/5 - \varepsilon$ was shown for 1-in-E3SAT in [6]. We are unaware of any comprehensive prior investigation into the complexity of approximating 1-in-kSAT and its variants for larger k.

Our Results. For a maximization problem (such as a maximum constraint satisfaction problem), we define an α-approximation algorithm to be one which always delivers solution whose objective value is at least a fraction $1/\alpha$ of the optimum. A random assignment clearly satisfies at least a fraction $k2^{-k}$ fraction of constraints in any 1-in-kSAT instance. This gives a $2^k/k$-approximation algorithm, and no better approximation algorithm appears to be known for general k. We prove that, for sufficiently large k, it is NP-hard to approximate 1-in-EkSAT (and hence also the more general 1-in-kSAT) within a $2^{k-O(\sqrt{k})}$ factor. The result uses a gadget-style reduction from the general Max kCSP problem, but the analysis of the reduction uses a "random perturbation" technique which is a bit different from the standard way of analyzing gadget-based reductions.

The easiest of the problems we consider, namely 1-in-EkHS, has a simple e-approximation algorithm. We prove that this is the best possible — obtaining an $(e-\varepsilon)$-approximation is NP-hard, for every $\varepsilon > 0$ (for large enough k). Using the algorithm for 1-in-EkHS, one can give an $O(\log k)$-approximation algorithm for 1-in-kHS. (Recently, in [4], a hardness result for approximating 1-in-kHS within a factor $\log^\sigma n$ has been shown. Here n is the size of the universe, k is allowed to grow with the input, and $\sigma > 0$ is an absolute constant.)

The $2^{k-O(\sqrt{k})}$ inapproximability factor for general 1-in-kSAT says that algorithms that are substantially better than picking a random assignment do not exist, assuming P $\neq$ NP. For satisfiable instances of 1-in-kSAT, however, we are able to give an e-approximation algorithm. Specifically, when given a 1-in-kSAT instance for which an assignment that satisfies *every* clause exists, the algorithm finds an assignment that satisfies at least a fraction $1/e$ of the constraints. This is again the best possible, since our $(e - \varepsilon)$-inapproximability result holds for satisfiable instances of 1-in-EkHS.

Our results highlight the following peculiar behavior of 1-in-kSAT which is unusual for constraint satisfaction problems. The 1-in-kSAT problem becomes much easier to approximate when negations are not allowed (the 1-in-kHS variant, which has a $O(\log k)$-approximation algorithm), or when restricted to satisfiable instances (which has an e-approximation algorithm).

A pricing problem. The 1-in-kHS problem is related to a natural optimization problem concerning pricing that was recently considered in [7]. Consider the following pricing question: there is a universe of n items, each in unlimited supply with the seller. There are m customers, and each customer is single-minded and wants to buy a precise subset of at most k items (and this subset is known to the seller). Each customer values his/her subset at one dollar, and thus will buy the subset if and only if it costs at most one dollar in total. The goal is set prices to the items that maximizes the total revenue. If the prices are all either 0 or 1 dollars, then this problem is exactly 1-in-kHS. But the seller could price items in cents, and thereby possibly generating more revenue. However, it can be shown that the optimum with fractional prices is at most e times that with $0,1$-prices (this follows from Lemma 6). Therefore, this problem admits a constant factor approximation (with approximation ratio independent of k) if and only if 1-in-kHS admits such an algorithm.

The Problem of Constructing "Ad-Hoc Selective Families." Another application of the 1-in-kHS problem is to the computation of *ad-hoc selective families*. An (n, h)-*selective family* is a combinatorial object defined and studied in [2] to deal with a broadcast problem in a radio network of unknown topology. An (n, h)-selective family is a collection $\mathcal{S}$ of subsets of $[n]$ such that for every set $F \subseteq [n]$ such that $|F| \leq h$ there is a set $S \in \mathcal{S}$ such that $|F \cap S| = 1$. More generally, a collection $\mathcal{S}$ of subsets of $[n]$ is *ad-hoc selective* for a collection $\mathcal{F}$ of subsets of $[n]$ if for every set $F \in \mathcal{F}$ there is a set $S \in \mathcal{S}$ such that $|S \cap F| = 1$ (in such a case, we say that S *selects* F). The notion of ad-hoc selective family, introduced in [3], has applications to the broadcast problem in radio networks of *known* topology. Here the computational problem of interest is, given a family $\mathcal{F}$ (that is related to the topology of the radio network) to find a family $\mathcal{S}$ of smallest size that is ad-hoc selective for $\mathcal{F}$: the family $\mathcal{S}$ determines a schedule for the broadcast on the radio network and the time needed to realize the broadcast depends on the number of sets in the family $\mathcal{S}$. Clementi et al. [3] observe that given an approximation algorithm for 1-in-kHS one can get an approximation algorithm for the ad-hoc selective family problem as follows. Think of $\mathcal{F}$ an instance of 1-in-kHS over the uniform $[n]$, then an approximation algorithm for 1-in-kHS will find a set S that "selects" a large number of sets in $\mathcal{F}$; then one deletes those sets from $\mathcal{F}$ and repeats the above process. In order to optimize this process, Clementi et al. [3] introduce the idea of dividing $\mathcal{F}$ into sub-families of sets having approximately the same size, and our $O(\log k)$ approximation algorithm for 1-in-kHS is based on a similar idea.

2 Approximation algorithms

In this section we present a randomized algorithm that delivers solutions within factor $1/e$ of the optimum solution for 1-in-EkHS. We also present a randomized algorithm that given an instance of 1-in-kSAT that is satisfiable, finds an assignment that satisfies an expected fraction $1/e$ of the clauses. Both these algorithms are the best possible in terms of approximation ratio, for large k, as we show in Section 3.

2.1 Approximation algorithm for 1-in-EkHS

For the simplest variant 1-in-EkHS, there is a trivial e-approximation algorithm.

Theorem 3 *For every integer $k \geq 2$, there is a polynomial time e-approximation algorithm for* 1-in-EkHS*. The claim holds also when k is not an absolute constant but an arbitrary function of the universe size.*

Proof: Set each variable to 1 independently with probability $1/k$. The probability that a clause of k variables has exactly one variable set to 1 equals $k \cdot \frac{1}{k}(1-1/k)^{k-1} = (1-1/k)^{k-1} \geq 1/e$. Therefore the expected fraction of clauses satisfied by such a random assignment is at least $1/e$. The algorithm can be derandomized using the method of conditional expectations. Note that we did not use the exact value of k in the above argument, only that all sets had size k. ■

2.1.1 Algorithm for 1-in-kHS.

We now consider the case when not all sets have exactly k elements. For this case we do not know any way to approximate within a factor that is independent of k.

Theorem 4 *There exists $c > 0$ such that for every integer $k \geq 2$, there is a polynomial time $c \log k$-approximation algorithm for* 1-in-kHS*. The claim holds even when k is not a constant but grows with the universe size.*

Proof: Let m be the number of sets in the 1-in-kHS instance. We partition the sets in the 1-in-kHS instance according to their size, placing the sets of size in the range $[2^{i-1}, 2^i)$ in collection $\mathcal{F}_i$ partition, for $i = 1, 2, \ldots, \lceil \log k \rceil$. Pick the partition that has a maximum number of sets, say $\mathcal{F}_j$, breaking ties arbitrarily. Clearly, $|\mathcal{F}_j| \geq \frac{m}{\lceil \log k \rceil}$. Set each element to 1 with probability $1/2^j$. Fix a set set in $\mathcal{F}_j$ that has x elements, $2^{j-1} \leq x < 2^j$. The probability that it has exactly one variable set to 1 equals $\frac{x}{2^j}\left(1 - \frac{1}{2^j}\right)^x$, which is easily seen to at least $\frac{1}{2e}$. Thus, expected fraction of sets in $\mathcal{F}_j$ that have exactly one element set to 1 is at least $\frac{1}{2e}$. Therefore, we satisfy at least $\frac{m}{2e\lceil \log k \rceil}$ sets. The algorithm can again be derandomized using conditional expectations, and the argument holds for every k in the range $1 \leq k \leq n$, where n is the universe size. ■

Remark: Note that in the above algorithm, the upper bound on optimum we used was the total number of sets. With this upper bound, the best approximation factor we can hope for is $O(\log k)$. This is because there are instances of 1-in-kHS with m sets whose optimum is at most $O(\frac{m}{\log k})$. In fact, it can be shown that an instance obtained by picking an equal number of sets in each of the buckets at random will have this property with high probability.

2.2 Approximation algorithm for 1-in-kSAT with perfect completeness

So far we saw algorithms for the case when negations were not allowed. We now give an approximation algorithm for 1-in-kSAT for the case when the instance is in fact satisfiable. Later on, in Section 3, we will show that without this restriction, a strong inapproximability result for 1-in-kSAT holds. We will also show that the factor e is the best possible, even with this restriction.

Theorem 5 *For every $k \geq 2$, there is an e-approximation algorithm for satisfiable instances of* 1-in-kSAT*. The claim holds even when k is not an absolute constant but an arbitrary function of the number of variables.*

Proof: The approach is to use a linear programming relaxation, and apply randomized rounding to it to obtain a Boolean assignment to the variables. Let $(V, \mathcal{C})$ be an instance of 1-in-kSAT, where $V = \{x_1, x_2, \ldots, x_n\}$ is the set of variables and $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ is the set of clauses. For $j = 1, 2, \ldots, m$, define $\mathsf{pos}(C_j) \subseteq V$ to be those variables that appear positively (i.e., unnegated) in C_j, and $\mathsf{neg}(C_j)$ to be those variables that appear negated in C_j. Consider the linear program P with the following constraints in variables $p_1, p_2, \ldots, p_n$:

$$0 \le p_i \le 1 \quad \text{for } i = 1, 2, \ldots, n \ ,$$

$$\sum_{i: x_i \in \mathsf{pos}(C_j)} p_i + \sum_{i: x_i \in \mathsf{neg}(C_j)} (1 - p_i) = 1 \quad \text{for } j = 1, 2, \ldots, m.$$

The above program has a feasible solution. Indeed, let $a : V \to \{0, 1\}$ be an assignment that satisfies all clauses in $\mathcal{C}$. Then clearly $p_i = a(x_i)$ for $i = 1, 2, \ldots, n$ satisfies all the above constraints.

Solve the linear program (P) to find a feasible solution p_i^*, $1 \le i \le n$ in polynomial time. We need to convert this solution into an assignment $a^* : V \to \{0, 1\}$. We do this using randomized rounding. That is, for each x_i independently, we set

$$a(x_i) = \begin{cases} 1 & \text{with probability } p_i^* \\ 0 & \text{with probability } (1 - p_i^*) \end{cases}$$

Now consider the probability that a particular clause, say C_1, is satisfied. Let C_1 depend on r variables, $x_{i_1}, x_{i_2}, \ldots, x_{i_r}$. For $1 \le \ell \le r$, define $q_\ell = p_{i_\ell}^*$ if x_{i_ℓ} appears unnegated in C_1, and equal to $1 - p_{i_\ell}^*$ if x_{i_ℓ} appears negated in C_1. Then we have $q_1 + q_2 + \cdots + q_r = 1$. The probability that C_1 is satisfied equals

$$q_1(1-q_2)(1-q_3)\cdots(1-q_r) \ + \ (1-q_1)q_2(1-q_3)\cdots(1-q_r) + \cdots + \ (1-q_1)(1-q_2)\cdots(1-q_{r-1})q_r$$

By Lemma 6, this quantity is minimized when $q_1 = q_2 = \cdots = q_r = 1/r$, and thus is at least $(1-1/r)^{r-1} \ge 1/e$. Therefore the expected fraction of clauses satisfied by the randomized rounding is at least $1/e$, proving the theorem. ■

We now state the inequality that was used in the above proof. We present a proof of the inequality due to Chris Chang in Appendix A.

Lemma 6 *Let $r \ge 2$ and let $q_1, q_2, \ldots, q_r$ be non-negative integers that sum up to 1. Then the quantity*

$$q_1(1-q_2)\cdots(1-q_r) \ + \ (1-q_1)q_2(1-q_3)\cdots(1-q_r) \ + \ \cdots \ + \ (1-q_1)\cdots(1-q_{r-1})q_r$$

attains its minimum value when $q_1 = q_2 = \cdots = q_r = 1/r$. In particular, the quantity is at least $(1-1/r)^{r-1}$.

3 Inapproximability results

3.1 Factor $2^{\Omega(k)}$ hardness for 1-in-EkSAT

We now prove that, if P $\neq$ NP, then, for sufficiently large k, 1-in-EkSAT cannot be approximated within a factor of $2^{k-O(\sqrt{k})}$. The result uses a gadget-style reduction from the constraint satisfaction problem Max EkAND, but the analysis of the reduction uses a "random perturbation" technique which is a bit different from the standard way of analyzing gadget-based reductions.

3.1.1 Preliminaries.

We first define the Max EkAND problem. An instance of Max EkAND consists of a set of Boolean variables, and a collection of AND constraints of the form $l_1 \wedge l_2 \wedge \cdots \wedge l_k$ where each l_j is a literal. The goal is to find an assignment that satisfies a maximum number of the AND constraints.

Theorem 7 ([8]) *If* $\mathrm{P} \neq \mathrm{NP}$*, for every* $\varepsilon > 0$ *and for every integers* $q \geq 1$ *and* k *such that* $2q+1 \leq k \leq 2q+q^2$*, there is no* $(2^{k-2q}-\varepsilon)$*-approximation algorithm for* Max EkAND*. In particular, for every* $k \geq 7$*,* $\varepsilon > 0$*, there is no* $(2^{k-2\lceil\sqrt{k}\rceil} - \varepsilon)$*-approximate algorithm for Max* k*AND. Furthermore, if* $\mathrm{ZPP} \neq \mathrm{NP}$*, then for every* ε *there is a constant* c *such that there is no* $n^{1-\varepsilon}$*-approximate algorithm for Max* $(c \log n)$*AND.*

The furthermore part follows from the result of [8] plus the use of a randomized reduction described in [1].

3.1.2 Inapproximability Result.

We describe a reduction from Max EkAND to 1-in-EkSAT.

Lemma 8 *Suppose that there is a polynomial time* β*-approximation algorithm for* 1-in-EkSAT*, for some* $k \geq 3$*. Then there is an* $2ek\beta$*-approximate algorithm for* Max EkAND*.*

Proof: We describe how to map an instance φ_{AND} of Max EkAND into an instance φ_{oik} of 1-in-EkSAT. Let $l_1 \wedge \cdots \wedge l_k$ be a clause of the Max EkAND instance φ_{AND}. We introduce the constraints $\mathsf{ONE}(l_1, \mathsf{neg}l_2, \ldots, \mathsf{neg}l_k)$, $\mathsf{ONE}(\mathsf{neg}l_1, l_2, \ldots, \mathsf{neg}l_k)$, $\ldots$, $\mathsf{ONE}(\mathsf{neg}l_1, \mathsf{neg}l_2, \ldots, l_k)$, where $\mathsf{neg}\ell$ denotes the negation of the literal ℓ. If originally we had m AND constraints, now we have km constraints of 1-in-EkSAT.

Claim 1 *If there is an assignment that satisfies* t *constraints in* φ_{AND}*, then the same assignment satisfies at least* tk *constraints in* φ_{oik}*.*

> **Proof of claim:** For each clause $l_1 \wedge \cdots \wedge l_k$ that is true, then the k corresponding ONE constraints are satisfied. □

Claim 2 *If there is an assignment that satisfies* t' *constraints in* φ_{oik}*, then there is an assignment that satisfies at least* $t'/(2ek^2)$ *constraints in* φ_{AND}*.*

> **Proof of claim:** Take the assignment a that satisfies t' constraints in φ_{oik} and then randomly perturb it in the following way: for each variable independently, flip its value with probability $1/k$ and leave the value unchanged with probability $1 - 1/k$; we will estimate the expected number of clauses of φ_{AND} that are satisfied by this random assignment R.
>
> Let us look at a clause $l_1 \wedge \cdots \wedge l_k$ of φ_{AND} and the k corresponding clauses of φ_{oik}. We consider different cases, depending upon how many of the literals $l_1, l_2, \ldots, l_k$ are set to 1 by the assignment a.
>
> - If a satisfies all literals, then it satisfies all k clauses in φ_{oik} and the random assignment satisfies $l_1 \wedge \cdots \wedge l_k$ with probability at least $(1 - 1/k)^k \geq 8/27$, for $k \geq 3$.
> - If a satisfies $k - 2$ literals, then it satisfies 2 clauses in φ_{oik}, and the random assignment satisfies $l_1 \wedge \cdots \wedge l_k$ with probability at least $(1-1/k)^{k-2} \cdot k^{-2} \geq 1/(ek^2)$

- In all other cases, a satisfies none of the clauses corresponding to $l_1 \wedge \cdots \wedge l_k$ in φ_{oik}.

In each case, we have that the probability that the assignment satisfies $l_1 \wedge \cdots \wedge l_k$ is at least $1/(2ek^2)$ times the number of constraints corresponding to $l_1 \wedge \cdots \wedge l_k$ in φ_{oik} satisfied by a. If a satisfies t' constraints, it follows that the random assignment satisfies, on average, at least $t'/(2ek^2)$ clauses of φ_{AND}. □

Suppose now that we have a β-approximate algorithm for 1-in-EkSAT and that we are given in input an instance φ_{AND} of Max EkAND. Suppose that the optimum solution for φ_{AND} has cost opt. Then we construct an instance φ_{oik} of 1-in-EkSAT as described above, and give it to the approximation algorithm. By Claim 1, the optimum of φ_{oik} is at least $k \cdot \mathsf{opt}$, and so the algorithm will return a solution of cost at least $k \cdot \mathsf{opt}/\beta$. By Claim 2, we get a distribution over assignments for φ_{AND} that, on average, satisfies at least $\mathsf{opt}/(2e\beta k)$ constraints. An assignment that satisfies at least as many constraints can be found deterministically using the method of conditional expectations. ■

Theorem 9 *If* $\mathrm{P} \neq \mathrm{NP}$*, for every sufficiently large k and for every $\varepsilon > 0$, there is no polynomial time* $(2^{k-2\lceil\sqrt{k}\rceil}/(2ek) - \varepsilon)$*-approximation algorithm for* 1-in-EkSAT*. Furthermore, if* $\mathrm{ZPP} \neq \mathrm{NP}$*, for every ε there is a c such that there is a no polynomial time $n^{1-\varepsilon}$ approximation algorithm for* 1-in-$(c \log n)$SAT*.*

Proof: Follows from Theorem 7 and from the reduction of Lemma 8. For the "furthermore" part one should observe that the reduction does not increase the size of the input by more that a logarithmic factor. ■

3.2 Factor $e - \varepsilon$ hardness for 1-in-EkHS

In this section, we prove the following hardness result, which shows that the results of Theorems 3 and 5 are tight in terms of the approximation ratio.

Theorem 10 *For every $\varepsilon > 0$, for sufficiently large k, there is no polynomial time $(e - \varepsilon)$-approximation algorithm for* 1-in-EkHS*, unless* $\mathrm{P} = \mathrm{NP}$*. Furthermore, the result holds even when the instance of* 1-in-EkHS *is satisfiable.*

3.2.1 Multiprover systems.

Our proof will use the approach behind Feige's hardness result for set cover [5]. We will give a reduction from the multiprover proof system of Feige, which we state in a form convenient to us below. In what follows, we use $[m]$ to denote the set $\{1, 2, \ldots, m\}$.

Definition 11 (p-prover game) *For every integer $p \geq 2$ and a parameter u that is an even integer, an instance $\mathcal{I}$ of the p-prover game of size n is defined as follows.*

- *The instance consists of a p-uniform p-partite hypergraph $\mathcal{H}$ with the following properties:*
  - *[Vertices] The vertex set of $\mathcal{H}$ is given by $W = Q_1 \cup Q_2 \cup \cdots \cup Q_p$, where Q_i is the vertices in the i'th part (or prover), and $|Q_i| = Q = n^{u/2}(5n/3)^{u/2}$ for $i = 1, 2, \ldots, p$.*
  - *[Hyperedges] There are $R = (5n)^u$ hyperedges in $\mathcal{H}$, labeled by $r \in [R]$. Denote the r'th hyperedge, for $r \in [R]$, by $(q_{r,1}, q_{r,2}, \ldots, q_{r,p})$, where $q_{r,i} \in Q_i$ for $i \in [p]$.*

– *[Regularity] Each vertex in W belongs to precisely R/Q hyperedges.*

- *Define $B = 4^u$ and $A = 2^u$. For each $r \in [R]$, and $i \in [p]$, the instance consists of projections $\pi_{r,i} : [B] \to [A]$ each of which is (B/A)-to-1.*

The goal is to find a labeling $a : W \to [B]$ that "satisfies" as many hyperedges of $\mathcal{H}$ as possible, where we define the notion of when a hyperedge is satisfied below.

- *[Strongly satisfied hyperedges] We say that a labeling $a : W \to [B]$* strongly satisfies *a hyperedge $r \in [R]$ if*
$$\pi_{r,1}(a(q_{r,1})) = \pi_{r,2}(a(q_{r,2})) = \cdots = \pi_{r,p}(a(q_{r,p})) \ .$$

- *[Weakly satisfied hyperedges] We say that a labeling $a : W \to [B]$* weakly satisfies *a hyperedge $r \in [R]$ if at least two elements of the tuple*
$$\langle \pi_{r,1}(a(q_{r,1})),\ \pi_{r,2}(a(q_{r,2})),\ \cdots,\ \pi_{r,p}(a(q_{r,p})) \rangle$$
are equal.

Feige's result on the above p-prover games can be stated as follows.

Theorem 12 *There exists a constant $0 < c < 1$ such that for every $p \geq 2$ and all large enough u, given an instance $\mathcal{I}$ of the p-prover game with parameter u, it is NP-hard to distinguish between the following two cases, when it is promised that one of them holds:*

- YES INSTANCES: *There is a labeling that strongly satisfies* **all** *hyperedges.*
- NO INSTANCES: *No labeling weakly satisfies more than a fraction $p^2 c^u$ of the hyperedges.*

Note that difference between YES and NO instances is not just in the fraction of satisfied hyperedges, but also in how the hyperedge is satisfied (strong vs. weak).

3.2.2 Reduction to 1-in-EkHS.

The result of Theorem 10 clearly follows from Theorem 12 and the reduction guaranteed by the following lemma.

Lemma 13 *For every $\varepsilon > 0$, there exists a positive integer p such that for all large enough even u the following holds. Let $k = 4^u$. There is polynomial time reduction from p-prover games with parameter u to* 1-in-EkHS *that has the following properties:*

- *[Completeness]: If the original instance of the p-prover game is a* YES *instance (in the sense of Theorem 12), then the instance of* 1-in-EkHS *produced by the reduction is satisfiable.*
- *[Soundness]: If the original instance of the p-prover game is a* NO *instance (in the sense of Theorem 12), then no assignment satisfies more than a fraction $(1/e + \varepsilon)$ of the constraints in the instance of* 1-in-EkHS *produced by the reduction.*

Proof: We begin with describing the reduction. Suppose we are given an instance $\mathcal{I}$ of the p-prover game with parameter u. In the following, we will use the notation and terminology from Definition 11. We define an instance of 1-in-EkHS on the universe

$$U \stackrel{\text{def}}{=} \{(i, q, a) \mid i \in [p], q \in Q_i, a \in [B]\} \ .$$

Thus the universe simply corresponds to all possible $\langle$vertex, label$\rangle$ pairs. The collection of sets in the instance is given by $\{S_{r,x}\}$ as r ranges over $[R]$ and x over $\{1, 2, \ldots, p\}^A$, where

$$S_{r,x} \stackrel{\text{def}}{=} \{(i, q_{r,i}, a) \in U \mid x_{\pi_{r,i}(a)} = i\} \ .$$

Note that the size of each $S_{r,x}$ equals $B = 4^u = k$. This is because $S_{r,x} = \bigcup_{j\in[A]}\{(i, q_{r,i}, a) \mid i = x_j,\ \pi_{r,i}(a) = j\}$, and for each $j \in [A]$, there are precisely B/A elements $a \in [B]$ such that $\pi_{r,i}(a) = j$ (since the projections are (B/A)-to-1).

Let us first argue the completeness (this will also help elucidate the rationale for the choice of the sets $S_{r,x}$).

Claim 3 (Completeness) *Let $a : W \to [B]$ be an assignment that strongly satisfies all hyperedges of $\mathcal{I}$. Consider the subset $C = \{(i, q, a(q)) \mid i \in [p],\ q \in Q_i\}$. Then $|C \cap S_{r,x}| = 1$ for every $r \in [R]$ and $x \in [p]^A$.*

Proof of claim: Since a strongly satisfies every hyperedge $r \in [R]$, we have $\pi_{r,1}(a(q_{r,1})) = \pi_{r,2}(a(q_{r,2})) = \cdots = \pi_{r,p}(a(q_{r,p}))$, and let $j_r \in [A]$ denote this common value. Also let $i_r = x_{j_r} \in [p]$. Then it is not hard to check that $C \cap S_{r,x} = \{(i_r, q_{r,i_r}, a(q_{r,i_r}))\}$. □

Claim 4 (Soundness) *Suppose that some $C \subseteq U$ satisfies $|C \cap S_{r,x}| = 1$ for at least a fraction $1/e + \varepsilon$ of the sets $S_{r,x}$. Then, provided $p \geq 1 + 3/(e\varepsilon)$ and $c^u < \varepsilon/(2p^8)$, $\mathcal{I}$ is not a* No *instance.*

Proof of claim: For each $i \in [p]$ and each vertex $q \in Q_i$, define $\mathcal{A}_q = \{a \in [B] \mid (i, q, a) \in C\}$, i.e,, $\mathcal{A}_q$ consists of those labels for q that the subset C "picked". We will later use the sets $\mathcal{A}_q$, $q \in W$ to prove that a good labeling exists for $\mathcal{I}$.

Call $r \in [R]$ to be *nice* if at least a fraction $(1/e + \varepsilon/2)$ of the sets $S_{r,x}$, as x ranges over $[p]^A$, satisfy $|C \cap S_{r,x}| = 1$. By an averaging argument, at least a fraction $\varepsilon/2$ of $r \in [R]$ are nice.

Let us now focus on a specific r that is nice. Define

$$D_r = \{(i, b) \mid i \in [p],\ b \in [A],\ (i, q_{r,i}, a) \in C \text{ for some } a \text{ s.t. } \pi_{r,i}(a) = b\} \ .$$

That is D_r consists of the projections of the assignments in $\mathcal{A}_{q_{r,i}}$ for all vertices $q_{r,i}$ belonging to hyperedge r. Let $|D_r| = M$ and $(i_1, b_1), (i_2, b_2), \ldots, (i_M, b_M)$ be the elements of D_r.

Now if $|C \cap S_{r,x}| = 1$, then *exactly one* of the events $x_{b_j} = i_j$ must hold as j ranges over $[M]$. Since r is nice we know that the fraction of such $x \in [p]^A$ is at least $(1/e+\varepsilon/2)$. We consider the following cases.

Case A: $M > p^3$. Then there are at least $M' \stackrel{\text{def}}{=} \lceil M/p \rceil > p^2$ distinct values among $b_1, b_2, \ldots, b_M$. For definiteness, assume that $b_1, b_2, \ldots, b_{M'}$ are distinct. If exactly one of the events $x_{b_j} = i_j$ holds as j ranges over $[M]$, then certainly at most one j in the range

$1 \leq j \leq M'$ satisfies $x_{b_j} = i_j$. The fraction of $x \in [p]^A$ for which $\Big|\{j \mid j \in [M'],\ x_{b_j} = i_j\}\Big| \leq 1$ is at most

$$\Big(1 - \frac{1}{p}\Big)^{M'} + M'\Big(1 - \frac{1}{p}\Big)^{M'-1} \leq e^{-(M'-1)/p}(M'+1) \leq e^{-p}(p^2+2) < 1/e$$

for $p \geq 10$. This contradicts that fact that r is nice. Therefore this case cannot occur and we must have $M \leq p^3$.

Case B: $M \leq p^3$ and all the b_j's are distinct. Clearly, the fraction of $x \in [p]^A$ for $x_{b_j} = i_j$ for exactly one choice of $j \in [M]$ is precisely $\frac{M}{p}\big(1 - 1/p\big)^{M-1}$. Now we bound this quantity as follows:

$$\begin{aligned}
\frac{M}{p}\big(1 - 1/p\big)^{M-1} &= \frac{p}{p-1}\frac{M}{p}\big(1-1/p\big)^M \\
&\leq \frac{p}{p-1}\frac{M}{p}e^{-M/p} \quad \text{(using } 1 - x \leq e^{-x} \text{ for } x \geq 0) \\
&\leq \frac{p}{p-1}\frac{1}{e} \quad \text{(using } xe^{-x} \leq 1/e \text{ for } x \geq 0) \\
&\leq \frac{1}{e} + \frac{\varepsilon}{3}
\end{aligned}$$

provided $p \geq 1 + 3/(e\varepsilon)$. Again, this contradicts that fact that r is nice. Therefore this case cannot occur either.

Therefore we can conclude the following: if r is nice, then $|D_r| \leq p^3$ and there exist $i_{r,1} \neq i_{r,2} \in [p]$ such that for some $b_r \in [A]$, $\{(i_{r,1}, b_r), (i_{r,2}, b_r)\} \subseteq D_r$.

Consider the following labeling a to W. For each $q \in W$, set $a(q)$ to be a random, uniformly chosen, element of $\mathcal{A}_q$ (if $\mathcal{A}_q$ is empty we set $a(q)$ arbitrarily). Consider a nice r. With probability at least $1/p^6$, we have

$$\pi_{r,i_{r,1}}\big(a(q_{r,i_{r,1}})\big) = \pi_{r,i_{r,2}}\big(a(q_{r,i_{r,2}})\big) = b_r$$

and thus hyperedge r is weakly satisfied by the labeling a.

In particular, there exists a labeling that weakly satisfies at least a fraction $1/p^6$ of the nice r, and hence at least a fraction $\frac{\varepsilon}{2p^6}$ of all $r \in [R]$. If u is large enough so that $p^2c^u < \frac{\varepsilon}{2p^6}$ (where c is the constant from Theorem 12), we know that $\mathcal{I}$ is not a No instance.

□

The completeness and soundness claims together yield the lemma. ■

4 Conclusions

The 1-in-EkSAT problem, while hard to approximate within a $2^{\Omega(k)}$ factor, becomes substantially easier and admits an e-approximation in polynomial time with either one of two restrictions: (i) do not allow negations (which is the 1-in-EkHS problem), (ii) consider satisfiable instances. Such a drastic change in approximability under such restrictions is quite unusual for natural constraint satisfaction problems (discounting problems which become polynomial-time tractable under these restrictions).

The following interesting question remains open:

- Does 1-in-kHS admit a polynomial time $o(\log k)$-approximation algorithm?[1]

Acknowledgments

We would like to thank Chris Chang for his proof of Lemma 6 and Ziv Bar-Yossef for useful discussions.

References

[1] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP's and non-approximability – towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

[2] B. S. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in unknown radio networks. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, pages 861–870, 2000.

[3] A. Clementi, P. Crescenzi, A. Monti, P. Penna, and R. Silvestri. On computing ad-hoc selective families. In *Proceedings of RANDOM'01*, 2001.

[4] E. Demaine, U. Feige, M. Hajiaghayi, and M. Salavatipour. Combination can be hard: approximability of the unique coverage problem. In *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2006. To appear.

[5] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[6] V. Guruswami. Query-efficient checking of proofs and improved PCP characterizations of NP. Master's thesis, MIT, 1999.

[7] V. Guruswami, J. Hartline, A. Karlin, D. Kempe, C. Kenyon, and F. McSherry. On profit-maximizing envy-free pricing. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2005.

[8] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[9] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing*, pages 216–226, 1978.

A Proof of Lemma 6

Proof: (due to Chris Chang) We will prove the lemma by an "equalization" method. Commonly, this is done by taking q_i, q_j which are unequal, replacing both of them by their average (so the sum remains the same), and arguing that the function value cannot increase. While this does not seem work here, a variant does: instead of equalizing two arbitrary terms, we equalize the largest and smallest q_i's at each step.

[1] A recent work [4] presents some evidence that perhaps such an algorithm does not exist.

Without loss of generality, let the largest and smallest values be q_1 and q_2 respectively, and let them be unequal. Initially the expression, call it A, is

$$\begin{aligned} A &= q_1(1-q_2)(1-q_3)\cdots(1-q_r) \;+\; (1-q_1)q_2(1-q_3)\cdots(1-q_r) \\ &\quad + (1-q_1)(1-q_2)\Big(q_3(1-q_4)\cdots(1-q_r) \;+\; \cdots \;+\; (1-q_3)\cdots(1-q_{r-1})q_r\Big)\ . \end{aligned}$$

To further simplify the notation, denote the sum of first two terms of the above by A_1, and the sum of the rest by A_2.

The new expression say B, obtained after replacing q_1 and q_2 by their average, is

$$\begin{aligned} B &= \frac{q_1+q_2}{2}\Big(1-\frac{q_1+q_2}{2}\Big)(1-q_3)\cdots(1-q_r) \;+\; \Big(1-\frac{q_1+q_2}{2}\Big)\frac{q_1+q_2}{2}(1-q_3)\cdots(1-q_r) \\ &\quad + \Big(1-\frac{q_1+q_2}{2}\Big)^2\Big(q_3(1-q_4)\cdots(1-q_r) \;+\; \cdots \;+\; (1-q_3)\cdots(1-q_{r-1})q_r\Big)\ . \end{aligned}$$

Similarly, denote the sum of the first two terms by B_1, and the sum of the rest by B_2. Our goal is to show that $A > B$. We have

$$\begin{aligned} A_1 &= (q_1+q_2-2q_1q_2)\cdot(1-q_3)\cdots(1-q_n)\ , \\ B_1 &= \Big(q_1+q_2-2\big((q_1+q_2)/2\big)^2\Big)\cdot(1-q_3)\cdots(1-q_n)\ , \end{aligned}$$

and hence

$$A_1 - B_1 = \frac{1}{2}(q_1-q_2)^2\cdot(1-q_3)\cdots(1-q_n) \tag{1}$$

Similarly,

$$\begin{aligned} A_2 &= (1-q_1-q_2+q_1q_2)\cdot(1-q_3)\cdots(1-q_n)\cdot\sum_{\ell=2}^{r}\frac{q_\ell}{1-q_\ell}, \\ B_2 &= \Big(1-q_1-q_2+\big(\frac{q_1+q_2}{2}\big)^2\Big)\cdot(1-q_3)\cdots(1-q_n)\cdot\sum_{\ell=3}^{r}\frac{q_\ell}{1-q_\ell}, \end{aligned}$$

and thus

$$B_2 - A_2 = \frac{1}{4}(q_1-q_2)^2\cdot(1-q_3)\cdots(1-q_n)\cdot\sum_{\ell=2}^{r}\frac{q_\ell}{1-q_\ell}\ . \tag{2}$$

Now, $\sum_{i=1}^{r} q_i = 1$, and q_1 is the largest of the q_i's. Therefore we have $q_\ell \le 1/2$, and thus $\frac{1}{1-q_\ell} \le 2$, for $3 \le \ell \le r$. Also since $\sum_{\ell=3}^{r} q_\ell \le 1$, we have

$$\sum_{\ell=3}^{r}\frac{q_\ell}{1-q_\ell} \le 2\ . \tag{3}$$

By (1), (2, and (3), we can conclude that

$$B_2 - A_2 = \frac{1}{2}(A_1-B_1)\sum_{\ell=3}^{r}\frac{q_\ell}{1-q_\ell} \le A_1 - B_1\ .$$

Thus $A - B = (A_1+A_2)-(B_1+B_2) = (A_1-B_1)-(B_2-A_2) \ge 0$, as desired. ■