PCP in Hardness of Approximation

Xingyou Song

Abstract

In this report, we study results from the past that have used the PCP theorem for hardness of approximation. These famous results include mostly, the $(1-1/e)$ bound for monotone submodular maximization and the SDP and its connection with max-CSP bounds. [1]

Contents

1 Submodular Bound

1.1 Preliminaries

Recall that a submodular set function is a boolean function $f : S \subseteq [n] \to \mathbb{R}$ that satisfies the diminishing returns property, i.e. $f(S \subseteq \{e\}) - f(S) \geq f(T \subseteq \{e\}) - f(T)$ for all $S \subseteq T$. A primary example of such a function is the coverage function, i.e. given that there exists a universe of ground elements $G = \{g_1, ..., g_m\}$ where $S_i \subseteq G$ for $i = 1, 2, ..., n$, then

$$f(S) = | \cup_{i \in S} U_i|$$

A submodular function is monotone if $f(S) \geq f(T)$ for all $T \subseteq S$. A very important problem in theory and machine learning is the monotone submodular maximization problem; i.e. given some budget k and a monotone submodular f, we want

$$S_{opt} = \arg\max_{|S|=k} f(S)$$

It is well known that the greedy algorithm gives a $(1 - 1/e)$ approximation bound, i.e. we can find a $|S_{poly}| = k$ in $poly(n)$ time such that $f(S_{poly}) \geq (1 - 1/k)^k f(S_{opt}) \geq (1 - 1/e)f(S_{opt})$.

However, it was only 20 years later that this constant $(1 - 1/e)$ was established as a hardness bound due to the PCP in [Fei98]; i.e. it is NP hard to guarantee an output that gives better than a $1 - 1/e$ approximation ratio. [2][3]

[1] I tried to read the Nash Equilibrium Inapproximability paper, but it was just too long

[2] [FMV11] Has a similar proof for non monotone functions.

[3] Although everybody uses this as a blackbox in the computational economics department, I think very few actually understand this proof, myself included before reading this.

1.1.1 Caveats

The [Fei98] paper establishes the inapproximability of the max-cover problem, which in turn establishes the general inapproximability of submodular functions. However, does not establish hardness of approximation for other, very different forms of submodular functions. For example, a completely different type of submodular function is of the form, given weights $w_1, ..., w_n$

$$f(S) = \max_{i \in S} w_i$$

This is still well known to have the same $1 - 1/e$ bound, even though it is not a covering problem.

Furthermore, it also does not establish approximation bounds for continuous extensions of submodular functions; i.e. the multilinear extension for a vector $x \in [0,1]^n$ is given by

$$g(x) = \mathbb{E}_{S \sim x}[f(S)]$$

where $S \sim x$ means each element i is chosen with probability x_i. [4]

1.2 Feige's PCP reduction

Feige's construction used the concept of a CNF formula with bounded clauses, from which he makes his own PCP construction (1-Round 2-Prover). Here, we introduce the problem of MAX 3SAT-5 from the concept of 3CNF-5 formulas, and use existing results to to show that this has an NP-hard distinguishability theorem.

Definition 1. *In the MAX 3SAT-5 problem, we are given a CNF formula with n variables and $5n/3$ clauses, in which each clause contains 3 literals, every variable appears in exactly 5 clauses, and a variable may only appear in a clause once.*

Given an arbitrary CNF formula, we can easily convert it to a 3CNF-5 formula: for each variable x that occurs b times, we may instead introduce variables $x_0, ..., x_{b-1}$ and replace those occurences of x, and then add in the new clauses $(x_i \vee \bar{x}_{i+1}), (\bar{x}_i \vee x_{i+1})$ for each $i = 0, ..., b-1$ which will give 5 instances of each x_i, satisfied when $x_0 = x_1 = ...x_{b-1}$. Since this reduction is clearly polynomial time and we've only scaled everything by a constant, then we can easily cite the theorem from Arora:

Theorem 1. *[ALM$^+$98] It is NP-hard to distinguish between total satisfiability and $(1-\epsilon)$ satisfiability for any 3-CNF formula*

This would imply the same theorem for our 3CNF-5 problem as well. The reason for the number 5 in particular is because it's the lowest number that wouldn't give us trouble.

1.2.1 1-Round 2-Prover

In the 1-Round 2-Prover protocol, the Verifier performs the following:

- Verifier queries a random clause, and Prover 1 provides an assignment to the 3 variables in that clause.
- Verifier queries a random variable from this clause.

To verify, the Verifier will:

1. Check the assignment given by the Prover 1 will satisfy the clause
2. Check that the variable assignment given by Prover 2 will be consistent with Prover 1's assignment

[4]These sorts of ideas, using simulated annealing for the multilinear extension for submodular functions, e.g. [GV11] are used

Theorem 2. *If $(1-\epsilon)$ fraction of the clauses can at most be satisfied, then the verification process will accept with probability at most $(1-\epsilon/3)$.*

Proof. Prover 2's assignment generates a total assignment to the problem. Hence, for a failed clause by Prover 1, he must've had at least one of the assignments to a variable differently from a clause he picked, and hence there is a failure probability of 1/3. $\square$

1.2.2 K-Prover System

[5] In the spirit of our class, Feige also cites parallel repetition to reduce the error in this scheme, using the paper by Raz [Raz98]. He briefly uses the fact that given the 1-Round 2-Prover scheme repeated in parallel ℓ times, then the error is $2^{-c(\epsilon)\ell}$ where $c(\epsilon)$ is a function of the original error.

However, he also introduces a completely new prover system - the k-prover system, which has become a very useful tool for various hardness of approximation bounds in economics, ex [CLLR15].

For a MAX 3SAT-5 instance ϕ, he first introduces the concept of a binary code, where there are k code words of length ℓ and $\ell/2$ one's, with pairwise distance at least $\ell/3$. $\ell = O(\log\log n)$.

Definition 2. *In Feige's k-prover system, the verifier selects ℓ random clauses independently, $C_1, ..., C_\ell$, and for each clause, uniformly and independently selects the variables in the clauses - $x_1, ..., x_\ell$. Each prover $P_i \in \{P_1, ..., P_k\}$ is assigned its corresponding codeword described above. For P_i, the clause requests $\{C_j\}$ he receives will be from the 1's locations (call this j) in his codeword, and the variable requests he receives will be from the 0-bit locations in his codeword.* [6]

Each prover will reply with this data - assignment to the variables in each clause, and assignments to the variables themselves.

It follows that the verifier will receive all of these assignments, and check for consistency among pairs of provers. Consistency here means, given two provers and their replies, all of their relevant clauses and variable assignments agree. He defines the following:

- Strong acceptance - Every pair of provers is consistent
- Weak Acceptance - At least one pair of provers is consistent.

For a guarantee, we have the following theorem:

Theorem 3. *If ϕ is satisfiable, then the verifier strongly accepts. Otherwise, if at most $(1-$ epsilon$)$-fraction of the clauses are satisfiable, then the verifier weakly accepts with probability at most $k^2 2^{-c(\epsilon)\ell}$*

Proof. The strongly acceptance guarentee is obvious. If $(1-\epsilon)$-fraction is satisfiable, then suppose the verifier weakly accepts with probability δ. A random prover pair will be consistent with probability at least δ/k^2 by the union bound. Since the code was made to be "spread out", then $\ell/6$ coordinates have differing queries (that is, clause and variable queries) from the two provers. They must have essentially $5\ell/6$ same pairs - if we assume in the worse case scenario they are exactly the same questions, then it follows that this is just a parallel repetition $\ell/6$ times, which implies $\delta/k^2 < 2^{c(\epsilon)\ell} \iff \delta < k^2 \cdot 2^{-c(\epsilon)\ell}$ $\square$

1.3 Set Cover

In set cover, the goal is to show that given a ground set $[n] = \{1, 2, ..., n\}$, with m subsets $S_i \subseteq [n] \forall i = 0, 1, ..., m$, set cover cannot be approximated within a ratio of $\ln n$.

What follows is a worst case-problem instance of set cover, based on a combinatorial construction.

[5] This tool has been used in many EC papers for submodularity - I just never understood them until now - yay!

[6] the reasoning for this is that the codewords are spread out - there is a correct mix of clauses and variable assignments that makes the consistency check more "uniform"

1.3.1 Partition System

[7] A partition system $B(m, L, k, d)$ is defined by:

- A ground set B of m points.
- L partitions $p_1, ..., p_L$, with each partition as a collection of k disjoint subsets of B.
- Any covering of the m points by subsets in different partitions requires at least d subsets

We may construct a partition system $B(m, L, k, d)$ such that:

- $L = (\log m)^c$
- $k < \ln(m/3) \ln \ln(m)$
- $d = (1 - o(1))k \ln m$

The construction is a simple probabilsitic one - for each element $i \in B$ and each p_j, randomly and uniformly choose where to put i in one of the k subsets of p_j.

The analysis is long and not too relevant here, but this is a **very** interesting example to keep in mind for future work - the worst case instances come from this sort of construction, which gives the question if we can get better bounds if we know that our problem instance is far from instances like these?

1.3.2 Set Cover Inapproximability

Here, we relate everything above from the PCP to set cover. When the verifier selects ℓ clauses and a distinguished variable for each clause, based on a random string r, we can generate a partition instance $B_r(m = n^{O(\ell)}, L = 2^\ell, k, d = (1 - o(1))k \ln m)$. There are $R = (5n)^\ell$ total possible random strings. Hence, we generate a set cover with $N = mR$ ground elements, with with R blocks of m separate partitions. Each subset in a partition corresponds to a prover - the subsets come from the k provers, where each prover may be asked one of $Q = n^{\ell/2}(5n/3)^{\ell/2}$ possible questions, implying kQ subsets total.

Now we provide the following equivalence:

Theorem 4. *If ϕ is satisfiable, then the $N = mR$ points can be covered by kQ subsets; otheriwse if $(1 - \epsilon)$ of the clauses can be satisfied, then the set requires at least $(1 - 2o(1))kQ \ln m$ subsets to be covered.*

The proof from this is long, but it translates back and forth between the ϕ to the PCP to this set cover instance.

1.3.3 Max Cover

The result from set cover will imply the $(1 - 1/e)$ bound for max-cover.

Proof. Suppose a polytime algorithm A can approximate max k cover with a ratio of $(1 - 1/e + \epsilon)$. Now suppose we have an instance of set cover. Then we can merely just try out $k = \{1, 2, .., n\}$ until we see that the max cover value is the entire set. Suppose k is actually the optimal. Each time we apply the max cover algorithm, we remove points that are covered (as well as the sets used), and hence to finish, we have that using at most ℓk sets, $(1/e - \epsilon)^\ell \leq 1/n \iff \ell = (1 - \delta) \ln n$ $\square$

[7] I realize now that recently inspired papers at STOC, e.g. have been inspired by these types of constructions, like [BRS16]

2 Constraint Satisfaction Problems and SDP

In this section, we give details of the PCP methods used to prove that Semi-Definite Programs achieve optimal bounds given by the PCP.

2.1 Preliminaries

In 1990's, the Semidefinite Programming relaxation was introduced for the MAX-CUT problem. It established an expected approximation bound of $\alpha_{GW} = 0.878$ as an improvement from a 0.5 approximation ratio from linear programming. What followed was the question of the existence of algorithms that perform better than the 0.878 ratio; do there exists more complicated algorithms, that give better approximations if we add in more reasonable time (but not to exponential time), like e.g. a better refinement between continuous and discrete space by using Sherali-Adams [SA90] or Lassere SDP's? [Las01]

In 2005, [KKMO07] proved that, assuming the Unique Games Conjecture, 0.878 was the maximum bound achievable. What followed in 2008 was [Rag08], which generalized this result to general constraint satisfaction problems - i.e. increasing computing power of SDP does not admit better approximation ratios.

The following methods will be the essential parts of the paper:

- Present the SDP for a constraint satisfaction problem.
- Present a dictatorship test for any assignment $\mathcal{F}$.
- Show that the SDP objective is equivalent to the dictatorship test value.
- Show the dictatorship test value is also equivalent to the value of a UGC instance.
- These two equivalences conclude that the SDP value is also related to the corresponding UGC value, and hence if we assume indistinguishability from good/bad instances from the UGC, then the SDP also has a hardness result.

The following are the definitions:

Definition 3. *A GCSP (Generalized Constraint Satisfaction Problem) is a sum of predicate "pay off" functions of the variable assignments, which output values in* $[-1, 1]$. *The goal of the max-GCSP is to maximize the sum of the pay-off functions. However, instead of total payoff, we wish to maximize expected payoff, which is total payoff divided by the number of payoff functions.*

We provide the following precise mechanics here:

For a GCSP, we have that $\Lambda = ([q], \mathbb{P}, k)$ where $[q] = \{0, 1, ..., q-1\}$ is an alphabet, $\mathbb{P} = \{P : [q]^t \to [-1, 1] | t \leq k\}$ is a set of payoff functions, and the number of inputs to a payoff function $P \in \mathbb{P}$ is called the arity. For example, MAX-CUT is GCSP.

An instance Φ of this GCSP is given by $\Phi = (\mathcal{V}, \mathbb{P}_V, W)$ where

- $\mathcal{V} = \{y_1, y_2, ..., y_m\}$ with variables taking values over $[q]$
- $\mathbb{P}_\mathcal{V}$ are the payoff functions with size at most k variables as input, and for notation, if $S = \{s_1, ..., s_t\} \subset \{1, ..., m\}^t$, then denote $y_{|S} = (y_{s_1}, ..., y_{s_t})$, and the output will be $P_S(y_{|S})$
- We have positive weights (which can be interpreted as a probability distribution) $\sum_{S \subset \mathcal{V}, |S| \leq k} w_S = 1$

We wish to maximize,

$$\mathbb{E}_{S \in W}\left[P_S(y|S)\right] = \sum_{S \subseteq [m], |S| \leq k} w_S P_S(y_{|S})$$

Using SDP(I), for a GCSP problem Λ, define $S_\Lambda(c)$ and $U_\Lambda(c)$:

Now we define the main SDP(I) program: [8]

$$\begin{aligned}
&\text{Maximize} \quad \sum_{S\in W} w_S \left(\sum_{\beta\in[q]^S} P_S(\beta(S)) X_{S,\beta} \right) \\
&\text{Subject to} \\
&\qquad v_{(i,c)} \cdot v_{(i,c')} = 0 \forall i \in [m], c \neq c' \in [q] \\
&\qquad \sum_{c\in[q]} v_{(i,c)} = I \quad \forall i \in [m] \\
&\sum_{\beta\in[q]^S, \beta(s)=c, \beta(s')=c'} X_{(S,\beta)} = v_{(s,c)} \cdot v_{(s',c')} \quad \forall S \in W, s, s' \in S, c, c' \in [q]
\end{aligned} \tag{1}$$

An explanation for this, is that we relax the assignment of a variable y_i in $[q]$, by producing variables $\{v_{(i,0)}, ..., v_{(i,q-1)}\}$. Furthermore, we assign $X_{(S,\beta)}$ where $S \subseteq [m]$ and $\beta \in [q]^S$ as an assignment to the set of variables defined by S. The inner products are used to be consistent with the probability distributions.

For notation, denote $FRAC(\Phi)$ be the SDP(I) value of the instance Φ, and $INT(\Phi)$ to be the integral value once we've used a rounding procedure *Round* for our instance - we give a brief introduction to this, mainly because of its usage of random Gaussian terms to aid with rounding:

2.1.1 Rounding Scheme for SDP

Once we have the fractional solutions from the SDP $(v_{i,c}, X_{S,\beta)})$, then we may use this rounding scheme:

- (To be added)

2.2 Dictator Test Value = SDP Value

Given an instance Φ, we can construct a dictator test $DICT_\Phi(f)$ for any q-ary function $f : [q]^R \to [q]$ (for any R). It proceeds by the following:

$\underline{DICT_\Phi(f) \text{ test}}$

- Pick sample a subset $S \subset \{1, ..., m\}$ using distribution W. Suppose our set is $S = \{s_1, ..., s_t\}$
- Since S is fixed, sample t times from the distribution formed by $X_{(S.\beta)}, \beta \in [q]^S$ to get the variable assignments $z_{s_1}, ..., z_{s_t}$
- For each $s \in S$ and $1 \leq j \leq R$, resample each coordinate $z_s^{(j)}$ with probability $(1 - \epsilon)$.
- Return Payoff : $P_{S\sim W}\left(f(\widetilde{z}_{s_1}), ..., f(\widetilde{z}_{s_t})\right)$

2.2.1 Completeness and Soundness equal to DICT value

[9] Here, we present the result that the SDP is approximately the same as the dictator test value: If $FRAC(\Phi)$ was the optimal solution from the SDP(I) of an instance Φ, then we have the following equivalence:

Theorem 5. *Completeness*$(DICT_\Phi) = FRAC(\Phi) - o(1)$ *and Soundness*$(DICT_\Phi) = \max_f Round_f(\Phi) + o(1) \leq INT(\Phi) + o(1)$

[8]Note that the number of variables used is exponential because we are defining every single set - For certain problems, we obviously can't do this in polynomial time if we don't know how to sample the distribution efficiently.

[9]We'll see that he precisely set up the SDP to match with the dictatorship test; the SDP is actually just the relaxed version of the dictatorship test.

Proof. When we arithmetize our dictatorship test, we can write it in the following algebraic way. Suppose we had an assignment $\mathcal{F} : [q]^R \to [q]$. Then when we write the arithmetization of the success of the dictatorship test, we get:

$$DICT_\Phi(\mathcal{F}) = \mathbb{E}_S \mathbb{E}_{\tilde{z}_{s_1},...\tilde{z}_{s_t}} \left[P_S \left(\mathcal{F}(\tilde{z}_{s_1}, ..., \mathcal{F}(\tilde{z}_{s_t})\right)\right] \tag{2}$$

If there was no noise, then note that this expectation is precisely equal to the value in 1.

Completeness: It follows that the maximal value of the SDP corresponds to the best we can hope for with the success probability of the dictatorship test (with a $o(1)$ unimportant detail).

Soundness: What we really want to prove is that in expectation if our $\mathcal{F}$ is far from a dictator function, then the expectation of the integral solution is close to the expectation of the fractional solution.

Remark: To gain some intuition, note that in the MAX-CUT SDP [KKMO07], we position the vectors representing the labellings of the vertices on a graph onto a hypersphere, optimize the sum of pairwise angular distances, and perform the rounding by taking a random hyperplane cut. If our instance was not distributed well (e.g. suppose the weight of a vertex was near infinity to give it the highest priority, which is like a dictator function), then the rounding scheme may give wildly different values at random (which gets amplified if we put these as input to a polynomial due to higher order moments).

Instead, if the instance was more balanced (the case being that the graph's vertices be projected onto a hypersphere and edge weights equal to pairwise angles, like a graph-like approximation to a sphere), then the cut-value from rounding always remains the same.

Firstly, we cite the following theorem, which says that for large random vectored input $z = (z^{(1)}, ..., z^{(t)})$, where $z^{(i)}$ can come from completely different distributions, a low degree multivariate polynomial inputting a high number (R) of copies of z, $P(z_1, ..., z_R)$, will have approximately the same valuation distribution as if we replaced the copies z with Gaussian vector copies $g = (g^{(1)}, ..., g^{(t)})$ where $g^{(i)}$ has the same mean and variance (i.e. limiting behavior) of $z^{(i)}$.

Essentially, this is a generalization of the central limit theorem for low degree polynomial functions, where our operations need not only be addition (in the case of the central limit theorem), but includes multiplication.

More formally, we have:

Theorem 6 (Invariance Principle)**.** *If H is a low degree polynomial with qR variables (q variables from each of the $z_1, ..., z_R$, and is far from a dictator function, then for any smooth function Ψ, we have that*

$$\mathbb{E}_{z_1,...,z_R}[\Psi(H(z_1, ..., z_R))] \approx \mathbb{E}_{g_1,...,g_R}[\Psi(H(z_1, ..., z_R))]$$

Let us we allow $H : \triangle_q^R \to \triangle_q$, where

$$H(z_1, ..., z_R) = \mathbb{E}[\mathcal{F}(\tilde{z}_1), ..., \mathcal{F}(\tilde{z}_R)|z_1, ..., z_R]$$

to represent the expectation after noise was added. The purpose of this is that it is low-degree smoothed version of $\mathcal{F}$, because its Fourier expansion is:

$$\mathcal{H}(z) = \sum_\sigma (1-\epsilon)^{|\sigma|} \hat{\mathcal{F}}_\sigma \prod_{i \in \sigma} z^{(i)}$$

We can see that high ϵ will have $(1-\epsilon)^{|\sigma|}$ very low for large products of coordinates σ, and hence $\mathcal{H}$ is close to low degree. Then it follows that from the invariance principle, if we let $\Phi = P$, the payoff function, then it follows that from 2.1.1 the expectation using integral rounding is equivalent to the expectation of the fractional solution. $\square$

2.2.2 Hardness Bounds UGC

Here, we present the key result, which will require the UGC:

Proof. Suppose we had a bipartite UGC instance $\Gamma = (\mathcal{X} \cup \mathcal{Y}, E, \Pi, \langle R \rangle)$ where $\mathcal{X}, \mathcal{Y}$ are the left and right side vertices, E is the set of edges, Π be the set of permutation functions across the edges, and $\langle R \rangle$ be the alphabet used in this case. We will construct an instance $\Phi(\Gamma) = (\mathcal{Y} \times [q]^R, \mathbb{P}, W)$. On a high level, we will generate an instance where the number of variables is the size of the long code for each element $w \in \mathcal{Y}$, and hence we have $\mathcal{Y} \times [q]^R$ variables in our problem. [10]

For formalism, if we if an assignment to all the variables, then we may index them by first the vertex $w \in \mathcal{Y}$ and then the specific part of the long code, i.e. we have a set of functions F^w:

$$F^w : [q]^R \to [q] \quad \text{for each } w \in \mathcal{Y}$$

which also implies a function that is the relaxation of F^w to admit a vector space output: $\mathcal{F}^w : [q]^R \to Conv(\triangle_q)$

$$\mathcal{F}^w(z) = e_{F^w(z)}$$

In the UGC, for each edge (v, w), there will be a permutation $\pi : [R] \to [R]$ in the direction from v to w. Then for an R-length vector input $z \in [q]^R$, we may define the operator $\pi^{-1}(z)$ as simply moving the coordinates of z according to π^{-1}. Then we may define, for $v \in \mathcal{X}$ a function $\mathcal{F}^v : [q]^R \to Conv(\triangle_q)$,

$$\mathcal{F}^v(z) = \mathbb{E}_{w \in N(v)}[\mathcal{F}^w(\pi_{vw}(z))]$$

[11]

To show the hardness result, we write the explicit algebraic formulation of 2.2.1: The verifier will need

During our test, the verifier will do the following:

- Pick random $v \in \mathcal{X}$
- (Oracle Subroutine) For any query of $\mathcal{F}^v(z)$, pick pick a random neighbor $w \in N(v)$, and then output $F^w(\pi_{vw}(z))$.
- Use the DICT test on $\mathcal{F}^v$ using the queries based on the oracle subroutine.

Note that we have one more expectation to handle because of this oracle subroutine, and hence when we perform the dictator test, the payoff will be in expectation:

$$\mathbb{E}_{v \in \mathcal{X}} \mathbb{E}_{z_i} \left[P \left(\mathbb{E}_{w_1 \in N(v)}[\mathcal{F}^{w_1}(\pi_{vw_1}(z_1)], ... \mathbb{E}_{w_t \in N(v)}[\mathcal{F}^{w_t}(\pi_{vw_t}(z_t)]) \right) \right] \tag{3}$$

which is essentially the expectation of the dictator function over a random v, i.e. $\mathbb{E}_{v \in \mathcal{X}}[DICT_\Phi(\mathcal{F}^v)]$

But note that the above equation (3) 2.2.2 is also the value of the UGC instance, fraction of satisfied edges (we will not formally prove it here, but this is intuitively true). Hence, the value of the dictatorship test is equal to the value of the fraction of satisfied edges in the UGC. Therefore, we may use 2.2.1 and see that since the soundness and completeness of the dictator test are equivalent to the fractional and integral solution values of the SDP. Then assuming the UGC, where it is NP-hard to distinguish between any fraction of edges satisfied and any fraction of edges not able to be satisfied, this implies that the SDP relaxation will also have a NP-hardness limit, and hence

[10] Not very new, as we've seen.

[11] An intuitive explanation of this function is that it is the "average" assignment of what the assignment of v should be, taken over all the neighbors $w \in N(v)$. Then it follows that a dictatorship test will decide, for a given assignment $\{F^w : w \in \mathcal{Y}\}$, how consistent this assignment will be towards a particular v. If the function was only based on one coordinate $z^{(j)}$, this implies that making the labelling of v as $j \in [R]$ will satisfy all the edge permutations when $\{F^w : w \in \mathcal{Y}\}$ is fixed.

Hence, it should follow that the acceptance probability of the dictatorship test is the number of satisfied edges in our UGC instance, and indeed this is the case - we'll use the UGC later to show optimality result.

the 1 is the best possible SDP; no higher relaxations such as Lassere or Sherali-Adams SDP will give a better result.

QED.

$\square$

References

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[BRS16] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. The power of optimization from samples. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4017–4025, 2016.

[CLLR15] Wei Chen, Fu Li, Tian Lin, and Aviad Rubinstein. Combining traditional marketing and viral marketing with amphibious influence maximization. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 779–796, New York, NY, USA, 2015. ACM.

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, July 1998.

[FMV11] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40(4):1133–1153, July 2011.

[GV11] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1098–1116, Philadelphia, PA, USA, 2011. Society for Industrial and Applied Mathematics.

[KKMO07] 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, April 2007.

[Las01] Jean B. Lasserre. *An Explicit Exact SDP Relaxation for Nonlinear 0-1 Programs*, pages 293–303. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[Rag08] Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 245–254, 2008.

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998.

[Rub17] Aviad Rubinstein. Settling the complexity of computing approximate two-player nash equilibria. *SIGecom Exch.*, 15(2):45–49, February 2017.

[SA90] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxation between the continuous and convex hull representations. *SIAM J. Discret. Math.*, 3(3):411–430, May 1990.