Relations between Average Case Complexity and Approximation Complexity

Extended Abstract

Uriel Feige
Faculty of Mathematics and Computer Science
Weizmann Institute, Rehovot 76100, Israel
feige@wisdom.weizmann.ac.il

ABSTRACT

We investigate relations between average case complexity and the complexity of approximation. Our preliminary findings indicate that this is a research direction that leads to interesting insights. Under the assumption that refuting 3SAT is hard on average on a natural distribution, we derive hardness of approximation results for min bisection, dense k-subgraph, max bipartite clique and the 2-catalog segmentation problem. No NP-hardness of approximation results are currently known for these problems.

1. INTRODUCTION

One way of coping with NP-hard combinatorial optimization problems is by relaxing the requirement for optimality of the solution, and settling for approximation algorithms. Another way is relaxing the requirement for worst case performance guarantees, and settling for algorithms that do well on average case instances. In this paper we explore connections between the two approaches.

The area of approximation algorithms was studied extensively in recent years. In particular, powerful techniques (the PCP theorem) have been developed that allow one to prove NP-hardness of approximation results. For most of the best known combinatorial optimization problems, there are approximation algorithms whose approximation ratios either match or are "in the same ball park" as the known hardness of approximation results. Still some exceptions remain. For example, it is currently not known how to prove NP-hardness of approximation results for min-bisection, dense k subgraph, or max bipartite clique, even though the best approximation ratios known for these problems are unbounded (grow as a function of the input size n). It has been suggested in the past that perhaps, rather than prove NP-hardness of approximation results, one can establish a weaker notion of hardness of approximation for these problems. For example, perhaps under some cryptographic complexity assumption (such as the existence of one-way functions, or the assumption that factoring is hard), one could establish that these problems are hard to approximate.

The results of this paper are related to the above approach. We assume that certain problems are hard on average (this is a typical requirement in cryptography), and based on these assumptions, derive hardness of approximation results for problems that so far resist NP-hardness of approximation results. For example, we show that if satisfiability is hard to refute on random 3CNF formulas with a sufficiently large number of clauses, then min-bisection cannot be approximated within a ratio better than 4/3.

1.1 Random 3SAT

A 3CNF formula is a collection of clauses, each containing three literals, where a literal is either a variable or its negation. A (Boolean) assignment to the variables is said to satisfy a clause if at least one of the literals in the clause is assigned the value true. A satisfying assignment is an assignment that satisfies all clauses in the formula. The problem of 3SAT, determining whether a 3CNF formula is satisfiable or not, is among the best known and most studied NP-complete problems. In particular, unless P=NP, every polynomial time algorithm fails to correctly determine satisfiablity on infinitely many 3CNF formulas.

In the current paper we consider the complexity of determining satisfiability of random 3CNF formulas. We assume that formulas are generated by the following random process. Given parameters n (for the number of variables) and m (for the number of clauses), each clause is generated independently at random by selecting the three literals that compose it independently at random. (There are several variants of the random 3CNF model that involve issues such as whether sampling is done with or without replacement, whether the number of clauses is fixed or is itself a random variable, whether each literal is required to appear the same number of times, and so on. These issues are not of great significance to this paper and we ignore them here.) We let Δ denote the ratio of clauses to variables in the formula. Namely, $\Delta = m/n$.

Random 3CNF's have been studied extensively. For surveys and additional references, see the recent special issue of TCS [22]. For the moment, let us remark that a simple (nonconstructive) probabilistic argument shows that when Δ is a large enough constant (satisfying $(7/8)^{\Delta} < 1/2$) then almost surely a random 3CNF formula is not satis-

STOC'02, May 19-21, 2002, Montreal, Quebec, Canada.

fiable. We ask whether there is a constructive version of the above fact. Namely, is there a polynomial time refutation algorithm that on most 3CNF formulas with $\Delta = m/n$ large enough announces that they are not satisfiable, and never falsely announces a satisfiable formula as nonsatisfiable. This question has been studied before [3, 12, 11], and refutation algorithms are known for the case when Δ is signifícatly larger than $\sqrt{n}$. In this paper we study the case when Δ is an arbitrarily large constant independent of n (and we let n grow keeping Δ fixed). For the sake of the current paper, we put forward the following hypothesis.

HYPOTHESIS 1. *Even when Δ is an arbitrarily large constant independent of n, there is no polynomial time algorithm that refutes most 3CNF formulas with n variables and $m = \Delta n$ clauses, and never wrongly refutes a satisfiable formula.*

The above hypothesis is consistent with the author's current knowledge. It is put forward so as to focus this paper and future research. Later in the paper we shall discuss the evidence there is in support (and against) this hypothesis. Based on Hypothesis 1, we can establish some new hardness of approximation results. It turns out that many of these hardness of approximation results can be proved under a weaker hypothesis, that we motivate and explain next.

Given a random 3CNF formula with a large enough number of clauses, we expect it not to be satisfiable. This is the typical case. The exceptional case is that it is satisfiable. Hence the problem of refuting random 3SAT can be viewed as the problem of designing a polynomial time algorithm that on every 3CNF formula either outputs *typical* or *exceptional*. The algorithm has one sided error with respect to input instances, in the sense that it never outputs *typical* on a satisfiable formula. The algorithm refutes most 3SAT instances (with sufficiently many clauses) in the sense that on at least 1/2 the 3CNF formulas, it outputs *typical*. (The constant 1/2 is arbitrary here. The choice of a particular constant over another does not affect the results of this paper. Formally, one can transform one constant to another by changing the value of $\Delta = m/n$. Details omitted.)

A simple probabilitic argument shows that when Δ is large enough, every assignment to the variables of a random 3CNF formula satisfies roughly $7m/8$ clauses. Hence a 3CNF formula can be viewed as exceptional not only when it is satisfiable, but also when there is an assignment that satisfies $(1-\epsilon)m$ clauses. A refutation algorithm becomes stronger if we forbid it to output *typical* also under this relaxed notion of exceptional. Hence the following hypothesis is weaker than Hypothesis 1.

HYPOTHESIS 2. *For every fixed $\epsilon > 0$, for Δ a sufficiently large constant independent of n, there is no polynomial time algorithm that on most 3CNF formulas with n variables and $m = \Delta n$ clauses outputs* typical*, but never outputs* typical *on 3CNF formulas with $(1-\epsilon)m$ satisfiable clauses.*

Hypothesis 2 has two main advantages over Hypothesis 1. First, it is weaker (Hypothesis 1 implies Hypothesis 2), and hence more likely to be correct. Second, Hypothesis 2 is more robust with respect to the model of generating random 3CNFs. (For example, the correctness of Hypothesis 1 may well depend on the question of whether literals within a clause are sampled with or without replacement, whereas this cannot effect the correctness of Hypothesis 2.) As an example for the differences between the two hypotheses, observe that with Hypothesis 1 one can output *typical* on an inverse polynomial fraction of 3CNF formulas just by considering the first eight clauses (which may happen to already give a nonsatisfiable instance), whereas with Hypothesis 2 one has to consider at least ϵm clauses in order to output *typical*.

1.2 Hardness of approximation

DEFINITION 1. *A computational problem is* random 3SAT-hard *(R3SAT-hard) if having a polynomial time algorithm for the problem contradicts Hypothesis 2.*

Technically, to show that a problem Π is R3SAT-hard one reduces random 3CNF formulas to instances of Π. As the kind of reduction used (many-to-one, Turing reduction, etc.) is not of great importance to this paper, the definition of R3SAT-hardness was given in general terms.

Remark: Definition 1 does not introduce a new complexity class (that of R3SAT-hard problems). By definition, if Hypothesis 2 is true, then every computational problem not solvable in polynomial time is R3SAT-hard. If Hypothesis 2 is false, then every computational problem is R3SAT-hard. Hence in both cases the class of R3SAT-hard problems coincides with a traditional complexity class (either that of the intractable computational problems, or that of all computational problems). There may be interest in considering restricted notions of R3SAT-hardness, such as the class of problems that are R3SAT-hard under Turing reductions, but this is beyond the scope of the current paper.

It is a triviality that Hypothesis 2 implies that 3SAT is R3SAT-hard to approximate within a factor better than 7/8. Thereafter, many of the known NP-hardness of approximation results can be derived also as R3SAT-hardness of approximation results by reduction from 3SAT. The more interesting fact is that we can establish R3SAT hardness of approximation results for problems that currently do not have NP-hardness of approximation results.

THEOREM 1. *Each of the following problems is R3SAT-hard to approximate within some constant factor: min bisection, dense k-subgraph, 2-catalog segmentation. The problem max complete bipartite subgraph is R3SAT-hard to approximate within a factor of n^{δ} where n is the number of vertices in the input graph, and $0 < \delta < 1$ is some constant.*

R3SAT hardness results are easier to establish than NP-hardnes results. The reason is that random 3CNF formulas have structure (or lack structure, depends on how one defines "structure"), and this can be used in proving the correctness of reductions.

1.3 Other functions on 3 variables

For 3SAT, the input is a collection of clauses of length three, and a clause is satisfied by an assignment if the OR of the three literals is 1. There are other possible functions to consider on three variables. In particular, we find the function AND very convenient for the reductions that prove Theorem 1.

Hypothesis 2 implies that it is hard to approximate max 3SAT within ratios better than 7/8 on random formulas. It is not hard to show that this implies that it is R3SAT-hard to approximate max 3AND within ratios better than 7/8 on random formulas. However, we prove a stronger theorem, that it is R3SAT-hard to approximate max 3AND within ratios better than 1/2 on random formulas. This is an interesting case where a hardness of approximation ratio is amplified from 7/8 to 1/2 by a reduction. We prove R3SAT hardness of approximation results of a similar spirit for random formulas over a whole class of functions (MAJORITY, XOR/LIN, and others) – see Theorem 2.

1.4 Outline

In Section 2 we discuss related work and how it effects the plausibility of Hypothesis 1 and 2. In Section 3 we prove Theorem 2, in particular showing that MAX 3AND is R3SAT-hard to approximate within a factor better than 1/2 on random formulas. In Section 4 we prove Theorem 1. The proofs in this section take an arbitrary 3AND formula ϕ and transform it into a bipartite graph G. It would be ideal if the quantity that we want to measure in G (such as the size of the bisection) would reflect the fraction of clauses satisfiable in ϕ. Unfortunately, this is not true in general, and there are certain worst case instances (formulas ϕ) that fool the reduction. However, an important observation (which proves Theorem 1) is that on average (namely, when ϕ is a random formula) the reduction does transform gaps in satisfiability to gaps in the desired graph property. In Section 5 we show that average case hardness assumptions other than Hypothesis 2 also lead to interesting hardness of approximation results. In Section 6 we discuss the significance of the results and some open questions.

A word on notation. In this manuscript, ϵ generally denotes a negligible quantity. In cases where we do not care about the exact value of ϵ, the same ϵ might denote different small quantities. Hence for example, it may happen that we use $\epsilon + \epsilon = \epsilon$.

2. RELATED WORK

The main theme of this paper is that average case hardness is a favorable starting point for deriving hardness of approximation results. The author believes that this was implicitly observed by other researchers, but is not aware of explicit references to this fact.

The satisfiablity of random 3CNF formulas has been the object of intense study (see [22] and references therein). It is known [10] that for every n there is a threshold Δ_n such that random formulas with n variables and significantly less than $\Delta_n n$ clauses are almost surely satisfiable, and formulas with significantly more than $\Delta_n n$ clauses are almost surely not satisfiable. It is known that $3 < \Delta_n < 4.6$, and experimental results suggest that $\Delta_n \simeq 4.2$, independently of n. It has been conjectured that at the threshold, determining satisfiablity of random 3CNF formulas is computationally hard. In order to prove this conjecture, it suffices to show that either proving satisfiability when $m/n = \Delta_l$ is hard, or refuting satisfiability when $m/n = \Delta_h$ is hard, for some $\Delta_l < \Delta_n$ or $\Delta_h > \Delta_n$. (Thereafter, hardness at Δ_n can be shown by either adding random clauses in the former case, or deleting random clauses in the latter case.) Our Hypothesis 1 claims that for some large enough Δ_h, refuting satisfiability when $m/n = \Delta_h$ is hard.

People familiar with experimental results on random 3CNFs may recall the experimental observation that though around the threshold 3SAT appears to be hard, as one gets further away from the threshold, determining satisfiability becomes easy in practice. This seems to refute Hypothesis 1. However, this is a wrong interpretation of the experimental results. “Easy” versus “hard” for the algorithms used in these experiments does not correspond to polynomial versus exponential running time. Rather, in both cases the running time is exponential, and the difference is only that the leading term in the exponent gets smaller as formulas get longer. The exponential behavior on random 3CNF formulas with Δ almost as large as $\sqrt{n}$ can be rigorously proved for a large class of algorithms, as a consequence of lower bounds on *resolution size* [5, 3, 4]

A natural approach to try to refute Hypothesis 1 is based on the observation that for large enough Δ, the fraction of satisfiable clauses in a random 3CNF formula is roughly 7/8. Hence if one could distinguish in polynomial time between satisfiable formulas and those in which only roughly $7m/8$ clauses are satisfiable then one could refute Hypothesis 1. But Hastad [14] shows that this problem is NP-hard. In a sense, hypotheses 1 and 2 say that to choose a 3CNF formulas on which MAX 3SAT is hardest to approximate, just choose one at random. This may be compared with the known fact that on worst case instances of MAX 3SAT, the best assignment one can choose efficiently is simply a random assignment. This seems somewhat related to the minimax theorem in game theory.

Recently, people studied for which values of Δ one can refute satisfiability of random 3CNF. It is not hard to see that when $\Delta > n$ the problem becomes solvable in polynomial time. (Consider the 3CNF subformula that includes all clauses containing the literal x_1. Remove x_1 to get a 2CNF formula, and use a polynomial time algorithm for 2SAT to show that it is not satisfiable. Then do the same with the 3CNF subformula containing the literal $\bar{x_1}$. The above argument can be extended to the case that $\Delta > n/\log n$, by considering all substitutions to $x_1, \ldots, x_{\log n}$.) Lower values of Δ are addressed using spectral techniques. In [12] it is shown how to refute random 4CNF formulas with Δ somewhat larger than n. In [11] random 3CNF formulas with Δ somewhat larger than $\sqrt{n}$ are reduced using resolution to 4CNF formulas with more than n^2 clauses, and then spectral techniques similar to those of [12] are used to refute satisfiability. Though the 4CNF formula is not random, the original randomness in the 3CNF formula suffices for the spectral techniques to work. (The proof becomes considerably more complicated, and is not included in [11].)

Algorithms that are stronger than plain use of spectral techniques involve semidefinite programming (SDP). The use of SDP for 3SAT as in [15] will NEVER refute a proper 3CNF formula in which no clause is shorter than 3. The author has tried a different SDP that does refute some 3CNF formulas. However, this SDP is not strong enough to refute Hypothesis 2. Details are sketched in the appendix.

The author does not have a strong opinion regarding the correctness of incorrectness of Hypotheses 1 and 2. However, Hypothesis 2 appears to be significantly more difficult to refute than Hypothesis 1. In addition to the evidence presented in the appendix, we note that the resolution rule (in its standard use) is not applicable in the context of Hypothesis 2

In this work, we prove that certain problems are R3SAT-hard to approximate. The problems addressed in this paper are min-bisection, max complete bipartite subgraph, dense k-subgraph, the 2-catalog segmentation problem. These problems will be defined in Section 4, and related work for each of these problems will be discussed after they are defined.

3. FUNCTIONS ON 3 VARIABLES

Let f be a Boolean function on 3 variables. Let t (for *true*) be the number of assignments that satisfy f (e.g., 1 for AND, 4 for XOR, 7 for OR). Let b (for *bias*) be the number of assignments with an odd number of 1's (or even number of 1's, whichever is larger) that satisfy f (e.g., 1 for AND, 4 for XOR, 4 for OR). There are 13 distinct functions of three variables (up to renaming or negation of variables) for which $2b > t$ (including AND, XOR, MAJORITY, OR, see [23] for example). A $3f$ formula is a formula in which each clause contains 3 literals, and a clause is satisfied by an assignment if the function f evaluates to *true* on that clause.

THEOREM 2. *For every function f on 3 variables and t and b as defined above, it is R3SAT-hard to distinguish between random instances of 3f formulas in which just over $t/8$ fraction of the clauses are satisfiable and those in which almost a $b/4$ fraction of the clauses are satisfiable. In particular, this implies that it is R3SAT-hard to approximate Max-3f within a factor better than $t/2b$. (Throughout this theorem, the random formulas have Δn clauses for some large enough Δ.)*

PROOF. Assume that we had an algorithm B for estimating the number of satisfied clauses in a random $3f$ formula within a ratio better than $t/2b$. We concentrate here on the special cases that f is either AND (for which $t = b = 1$) or XOR (also known as LIN, for which $t = b = 4$). Towards the end of the proof we explain how it needs to be modified so as to apply to MAJORITY (for which $t = 4$ and $b = 3$). Other functions f are treated similar to the way indicated for MAJORITY.

Let ϕ be a 3CNF formula with n variables and $m = \Delta n$ clauses, with Δ large enough. We show how B can be used to distinguish between the case that ϕ is *typical* (and hence only $(7/8 + \epsilon)$ satisfiable), and the case that ϕ is $(1 - \epsilon)$ satisfiable (*exceptional*).

We suggest the following algorithm for outputting *typical* on a large fraction of 3CNF formulas, but never on exceptional ones.

On average, each literal has $3\Delta/2$ appearances in ϕ. When Δ is large enough, standard bounds on large deviations show that with high probability, all but an ϵ fraction of the occurances of literals correspond to literals that appear between $(3/2 \pm \epsilon)\Delta$ times in ϕ. If this fails to hold on ϕ then output *exceptional*. If this does hold, observe that every assignment satisfies on average roughly $3/2$ variables per clause in ϕ. We shall use this fact later.

Construct three different multi-graphs (we allow parallel edges) from ϕ. The number of vertices in each graph is $2n$, one for each literal of ϕ. The edges in G_1 (G_2, G_3, respectively) are determined by scanning ϕ clause after clause, and for each clause of ϕ putting an edge between the vertices corresponding to the first and second literal of the clause (second and third literal, third and first literal, respectively).

On each of the graphs, use semidefinite programming as in [13] to give an upper bound on the size of the maximum cut. If this upper bound for any of the three graphs is above $(1/2 + \epsilon)m$, then output *exceptional*.

If the algorithm does not output *exceptional* up to this point, then consider ϕ to be a $3f$ formula (recall that here f is either AND or XOR), and run algorithm B on it. If B outputs that the number of clauses satisfiable (as $3f$ clauses) is below $(b/4 - \epsilon)m$, output *typical*. Otherwise, output *exceptional*.

We claim the following:

1. When Δ is large enough then on almost all formulas ϕ the final output of the algorithm is *typical*.
2. The algorithm never outputs *typical* on a 3CNF formula that is $(1 - \epsilon)m$ satisfiable.

To prove the claims observe that if ϕ is random, then each of the graphs G_i (for $i = 1, 2, 3$) is essentially a random graph with $2n$ vertices (one for each literal) and m edges (one for each clause). When m is sufficiently large, then a simple probabilistic argument shows that with high probability, the maximum cut in a random graph has at most $(1/2 + \epsilon)m$ edges. Moreover, as shown in [24], when the maximum cut in a graph has $(1/2 + \epsilon)m$ edges, then the upper bound output by the semidefinite program is also $(1/2 + \epsilon)m$ (for a different ϵ). Hence the algorithm reaches the stage that it treats ϕ as a $3f$ formula and runs B on it.

Let ψ be an arbitrary assignment to the n variables. Then we claim that viewing ϕ as a NAE 3SAT formula (a clause is satisfied if the literals are not-all-equal), at most $(3/4+\epsilon)m$ clauses are satisfied by ψ. Otherwise, taking a random clause from ϕ and a random pair of literals, there is probability greater than $\frac{2}{3}(\frac{3}{4} + \epsilon) \simeq \frac{1}{2} + \epsilon$ that ψ assigns different values to the literals. Hence at least in one of the three graphs G_i, the cut induced by the assignment ψ (a literal is in the left hand side if ψ assigns 0 to it and in the right hand side otherwise) has more than $(1/2 + \epsilon)m$ edges.

Now let ψ be an arbitrary assignment that satisfies $(1 - \epsilon)m$ clauses in ϕ (as a 3SAT formula). As the same ψ can satisfy at most $(3/4+\epsilon)$ clauses of ϕ as a NAE 3SAT formula, then it must satisfy at least $(1/4 - \epsilon)m$ clauses of ϕ as a 3AND formula. Moreover, as each literal appears roughly the same number of times in ϕ, it must be that ψ satisfies on average $3/2$ literals per clause in ϕ. As only ϵm clauses have no literal satisfied by ψ, it then follows that at least $(3/4 - \epsilon)m$ clauses have exactly one literal satisfied by ψ. Hence ψ satisfies $(1 - \epsilon)m$ clauses of ϕ as a 3XOR formula.

Observe that if ϕ is random, then only $(1/8+\epsilon)m$ clauses are satisfied as a 3AND formula, and only $(1/2+\epsilon)m$ clauses are satisfied as a 3XOR formula. This suffices to establish Theorem 2 for the functions f AND and XOR.

To prove the theorem for MAJORITY, flip all variables in each clause (changing positive literals to negated ones and vice versa). Now ψ as above satisfies the majority of literals in at least $(3/4 - \epsilon)m$ of clauses. Note that the flipped formula is still random, and hence the typical case is that every assignment satisfies the majority of literals in at most $(1/2 + \epsilon)m$ clauses. This establishes Theorem 2 for MAJORITY.

The same approach of flipping variables can be used to prove the theorem for the other 10 functions. In some cases one needs to flip only part of the variables (e.g., the first variable in each clause). Details omitted. □

The R3SAT hardness of approximation ratios in Theorem 2 match the best known NP-hardness results for every one of the 13 functions considered. (These ratios are proved in [14, 23].) In some cases, these bounds are best possible, as there are algorithms achieving these approximation ratios [23].

The new aspect in Theorem 2 is that it applies to random formulas and not just worst case instances, and that it unifies the known hardness of approximation ratios into a simple formula ($t/2b$), which is a natural consequence of the proof technique used here. (Of course, one should remember that Theorem 2 proves only R3SAT-hardness.)

For use in Section 4, we state explicitly the R3SAT-hardness result for MAX 3AND.

COROLLARY 1. *For every $\epsilon > 0$ and large enough Δ, it is R3SAT-hard to output* typical *on most 3AND formulas with n variables and $m = \Delta n$ clauses, but never output* typical *on 3AND formulas with more than $(1/4 - \epsilon)m$ satisfiable clauses.*

4. GRAPH PROBLEMS

Here we prove Theorem 1. We shall use the following technical proposition.

PROPOSITION 1. *For every $\epsilon > 0$ there is some $\Delta_\epsilon > 0$, such that for every $\Delta > \Delta_\epsilon$ and n large enough, with high probability the following holds. Every set of $(1/8+\epsilon)m$ clauses in a random 3CNF formula with $m = \Delta n$ clauses contains at least $n+1$ different literals.*

PROOF. Fix a set of S of n literals to be avoided. The probability that a random clause with three literals avoids these literals is 1/8. For large enough n and for Δ as in the proposition, standard bounds on large deviations imply that with probability greater than $1-2^{-3n}$, less than $(1/8+\epsilon)m$ random clauses avoid the set S. As there are only roughly 2^{2n} ways of choosing the set S, the union bound implies that no one of them is avoided by a set of $(1/8+\epsilon)m$ clauses. $\square$

4.1 Balanced Bipartite Clique

Input: An n by n bipartite graph G.
Output: A k by k complete bipartite subgraph of G.
Objective function: maximize k.

Bipartite cliques are natural objects to study in communication complexity models. They also come up in other contexts. For example, Woeginger (private communication) establishes a connection between approximating bipartite cliques and breaking the barrier of 2 for approximation of certain scheduling problems (minimizing average weighted job completion time on a single machine under precedence constraint).

The best approximation ratios known for the bipartite clique problem are of the order of $n/(\log n)^c$ for some constant c. No hardness of approximation result for this problem is known.

Related versions of the problem do not require the subgraph to be balanced. That is, the output may be a k_1 by k_2 complete bipartite subgraph with $k_1 \neq k_2$. In these versions, there are two natural objective functions. One is to maximize the number of edges $k_1 \cdot k_2$. The results of this section (with some modifications that are omitted) apply to this objective function as well. Another possible objective function is to maximize the number of vertices $k_1 + k_2$. This version is polynomial time solvable (using the fact that maximum independent set is polynomial time solvable in bipartite graphs).

Results of H. Simon [21] imply that it is NP-hard to approximate within a factor of n^δ (for some $\delta > 0$) the minimum number of complete bipartite subgraphs that together cover all edges of a given bipartite graph. It is an open question whether Simon's proof can be extended to show that approximating bipartite clique is hard. (Note: had the requirement in the covering problem been to cover all vertices rather than all edges, then a hardness of approximation result such as that of Simon would have easily implied a hardness of approximation result for bipartite clique.)

LEMMA 1. *For every $\epsilon > 0$, it is R3SAT-hard to approximate the bipartite clique problem within a factor of $1/2+\epsilon$. More specifically, it is R3SAT-hard to distinguish between the cases $k > (1/4-\epsilon)n$ and $k < (1/8+\epsilon)n$.*

PROOF. By reduction from random MAX 3AND. Given a random instance of MAX 3AND with n' variables and $m' = \Delta n'$ clauses, construct the following bipartite graph. Each side has $n = m'$ vertices, one for each clause of the 3AND formula. Two vertices on different sides are connected by an edge if both clauses can be satisfied simulateously (namely, unless there is a variable that appears in positive form in one of them and in negative form in the other).

Approximating bipartite clique within ratios better than stated in the lemma would allow one to distinguish between MAX 3AND formulas that are $(1/4-\epsilon)$-satisfiable, and those that are typical (and in particular only $(1/8+\epsilon)$-satisfiable).

If there are $(1/4-\epsilon)m'$ clauses satisfiable in the 3AND formula, then the vertices corresponding to these clauses are a k by k bipartite clique with $k = (1/4-\epsilon)n$.

If the 3AND formula is random, then by Proposition 1, every set of $(1/8+\epsilon)m'$ clauses contains at least $n'+1$ distinct literals. Hence every pair of such sets contains a variable that appears positively in one of the sets and negated in the other, excluding the possibility of a k by k complete subgraph with $k = (1/8+\epsilon)n$. $\square$

THEOREM 3. *For some $\delta > 0$, it is R3SAT-hard to approximate bipartite clique within a factor of $1/n^\delta$.*

Theorem 3 can be proved by applying the technique of derandomized graph products [2] to the graphs of Lemma 1. Details are omitted from this manuscript.

4.2 Min Bisection

Input: A graph G with n vertices, where n is even.
Output: A set S of $n/2$ vertices (a bisection).
Objective function: minimize the number of edges connecting S and $\bar{S}$ (the bisection width).

The best approximation ratio known for min bisection is $O(\log^2 n)$ [8]. There are no hardness of approximation results known for bisection. For the related problem of finding small separators, one can find a $(1/3, 2/3)$-separator of width at most $O(\log n)$ factor larger than the minimum bisection width [17].

THEOREM 4. *It is R3SAT hard to approximate min bisection within a ratio below 4/3.*

PROOF. By reduction from random MAX 3AND. Given a MAX 3AND formula with n' variables and $m' = \Delta n'$ clauses in which we want to distinguish between the case that at most $(1/8+\epsilon)m'$ clauses are satisfiable and the case that at least $(1/4-\epsilon)m'$ clauses are satisfiable, construct the following graph.

The left hand side (LHS) contains $2n'$ vertices, one for each literal. The right hand side (RHS) contains m' clusters, one for each clause, where each cluster is a clique of size $4m'$. In addition, the graph contains a clique of size $m'' = 4m'(1/2+2\epsilon)m'$. Hence the number of vertices in the graph is $n = 4m'(3/2+2\epsilon)m' + 2n'$. Note that we may assume that $(1/4-\epsilon)m'$ is an integer, which implies that the vertex set can be partitioned into two equal cardinality sets (a bisection) without cutting any of the clusters, nor the clique. (Place n' vertices and $(1/4-\epsilon)m'$ clusters and the clique on one side.)

In each cluster there is a unique vertex that is its "connecting vertex". Place an edge between a vertex that corresponds to a literal and the connecting vertex of a cluster if the literal is in the clause that corresponds to the cluster. These are called the "bipartite" edges.

In this graph, find a minimum bisection.

We note that the total number of bipartite edges in the graph is $3m'$. As there is a bisection cutting only bipartite edges, the minimum bisection neither cuts a cluster (as each such cut already includes $4m'-1$ edges), nor the clique. It follows that the minimum bisection contains exactly n' LHS vertices, and $(1/4-\epsilon)m'$ clusters, and the clique. Hence, for the rest of the discussion, it suffices to consider only the connecting vertices from each of the m' clusters, and we need to find a cut of minimum width that contains n' vertices from the LHS, and $(1/4-\epsilon)m'$ connecting vertices.

Observe that each connecting vertex has degree 3 (to the LHS). The average degree of LHS vertices is $3\Delta/2$. With high probability (over the choice of the random 3AND formula), only an ϵ fraction of the bipartite edges connect to LHS vertices of degree that deviates from $3\Delta/2$ by more than an ϵ fraction. If this condition fails to hold, we do not perform the reduction to bisection and allow the algorithm to output *exceptional* on the original 3AND formula. Absorbing an $O(\epsilon)$ error in our analysis, we shall assume that every LHS vertex has exactly the same degree $d = 3\Delta/2$.

When the 3AND formula has $(1/4-\epsilon)m'$ satisfiable clauses, we pick the set S to contain the clusters corresponding to these clauses, the clique, and the n' literals corresponding to the assignment consistent with these clauses. The only edges cut by this bisection connect the satisfying literals to unsatisfied clauses. The number of bipartite edges within the set S is $3(1/4-\epsilon)m'$. The sum of degrees of the satisfied literals is $3n'\Delta/2 = 3m'/2$. Hence the width of the bisection is $3(1/4+\epsilon)m'$.

In a random 3AND formula, we still need one side of the cut to contain n' vertices and $(1/4-\epsilon)m'$ clusters and the clique. This set of n' literals has at most $(1/8+\epsilon)m'$ of these clauses 3-connected to it (by Proposition 1) and the other $(1/8-2\epsilon)m'$ clauses are 2-connected to it. Hence the width of the cut is at least

$$3m'/2-3(1/8+\epsilon)m'-2(1/8-2\epsilon)m'+(1/8-2\epsilon)m' = (1-\epsilon)m'$$

The ratio between bisection width of the two cases is arbitrarily close to 4/3.

Summarizing, if one could approximate bisection within a factor better than 4/3, then one would have the following procedure for recognizing typical 3AND formulas (that are only $1/8+\epsilon$ satisfiable).

- Perform the reduction to bisection as above.
- If the degree constraints on the LHS are not satisfied, output *exceptional*.
- Run the approximation algorithm for bisection, and output *typical* if the output excludes a bisection of width $3(1/4+\epsilon)m'$.

□

4.3 Dense k-subgraph

Input: A graph G with n vertices, and a parameter k.
Output: A vertex induced subgraph on k vertices.
Objective function: maximize the number of edges in the subgraph.

The dense k-subgraph problem can be approximated within a factor of $O(n^{\delta})$ for some $\delta < 1/3$ [7]. No hardness of approximation result is known for it. When k is fairly large, e.g. $k = n/2$, the known approximation ratio is somewhat above n/k [9]. We remark that a constant factor hardness of approximation for the dense k-subgraph problem on regular graphs when $k = \Omega(n)$ implies constant factor hardness of approximation for bipartite clique and for min bisection (via reductions that are omitted from this manuscript).

THEOREM 5. *The dense k-subgraph problem is R3SAT hard to approximate within some constant $\rho < 1$.*

PROOF. Given a 3AND formula with n' variables and m' clauses, we construct the following bipartite graph G. On the LHS we have $2n'$ vertices corresponding to the literals. On the RHS we have m' clauses corresponding to the clauses. We assume that $n' = m'/16$. This assumption can be made without loss of generality, as if it does not hold, either vertices corresponding to clauses or to variables can be duplicated and the reduction still works. Details are omitted from this manuscript. (The choice of multiplier 1/16 is somewhat arbitrary here and can be optimized to give tighter results.)

An edge connects a LHS vertex to a RHS vertex if the corresponding literal is in the corresponding clause.

We choose $k = n' + (1/4-\epsilon)m'$.

If the number of satisfiable clauses in the 3AND formula is $(1/4-\epsilon)m'$, we have a solution that takes the vertices corresponding to these clauses and to the literals of the underlying assignment, obtaining a subgraph with a $(1/4-\epsilon)$ fraction of the edges of G.

If the input is a random 3AND formula with only $(1/8+\epsilon)m'$ satisfiable clauses, then the densest k-subgraph in G is significantly less dense. Every n' by $(1/4-\epsilon)m'$ subgraph contains only a $(1/8+\epsilon)+(2/3)(1/8-2\epsilon) < 5/24$ fraction of the edges. Other k-subgraphs have $(1+\beta)n'$ LHS vertices and $(1/4-\epsilon-\beta/16)m'$ RHS vertices, for some β. Unless $|\beta|$ is bounded away from 0, it can be shown that the number of edges in the subgraph does not significantly exceed a 5/24 fraction of the edges of G. Hence we assume that $|\beta|$ is bounded away from 0.

If $\beta > 0$ then the proof follows from the fact that the fraction of edges in the subgraph is at most $(1/4 - \epsilon - \beta/16)m'$.

If $\beta < 0$, then the number of RHS vertices (clauses) increases, but number of LHS vertices (literals) decreases at a relatively higher rate (by the relation $n' = m'/16$). We have $(1+\beta)n'$ literals. The fraction of clauses containing only these literals is at most $(1+\beta)^3/8 + \epsilon$ (via a proof similar to that of Proposition 1). The rest of the clauses in the subgraph have degree at most 2. Ignoring ϵ terms, by taking $0 > \beta > -1$, the fraction of edges lost is $\frac{1}{8}(3\beta + 3\beta^2 + \beta^3)$ and the fraction of edges gained (by having more clauses) is $-\frac{\beta}{16} \cdot 2$, which amounts to no net gain. (The function $2\beta + 3\beta^2 + \beta^3 = \beta(1+\beta)(2+\beta)$ is negative for $0 > \beta > -1$.) □

4.4 The 2-catalog problem

Input: An n by n bipartite graph and a parameter r.
Output: Two (not necessarily disjoint) subsets S_1 and S_2 of size r of the RHS.
Objective function: maximize

$$\sum_{v \in LHS} \max[|E(v, S_1)|, |E(v, S_2)|].$$

The 2-catalog problem was introduced in [16]. The LHS vertices represent clients and the RHS vertices represent items. One has to construct two catalogs, each with r items, and send to each client one of the catalogs. The objective is to maximize the number of items of interest that a client sees in the catalog that he/she receives, summed over all clients. An edge between a client and an item in the bipartite graph signifies that the client is interested in the item. The catalog problem was also used to model certain problems in coding theory [19].

The 2-catalog problem is NP-hard. It can easily be approximated within a factor of 1/2 (e.g., by picking just one optimal catalog of size r and sending it to all the clients), and no better approximation ratio is known for it, except for special cases (such as when $r = n/2$, see [6]). There is no hardness of approximation result known for this problem. (There was a manuscript circulated with a claim of a factor 1/2 NP-hardness of approximation result, but this claim was retracted. The reduction below is partly based on this failed attempt, but we only claim R3SAT hardness, and within a factor different than 1/2.)

Theorem 6. *For some $\rho < 1$, it is R3SAT-hard to approximate the 2-catalog segmentation problem within a ratio better than ρ.*

Proof. Consider a 3AND formula with n' variables and m' clauses. First transform it to a $3k$-AND formula with n' variables and $2(m')^k$ clauses of length $3k$, where k is a large enough constant. The first $(m')^k$ clauses are obtained as ANDs of sequences of k clauses from the 3AND formula. For simplicity in the analysis, we assume that the $3k$ literals in a clause are always distinct. (We can remove clauses in which this does not hold, and the influence of this on the analysis will be negligible.) The other $(m')^k$ clauses are obtained by complementing each literal in the first $(m')^k$ clauses.

Now construct a bipartite graph as follows. The LHS contains $2n'$ vertices, one for each literal. The RHS contains $2(m')^k$ vertices, one for each clause of the $3k$-AND formula. A LHS vertex (literal) is connected to all RHS vertices (clauses) that contain the corresponding literal. We let $r = ((1/4 - \epsilon)m')^k$.

Consider first the case that there is an assignment ϕ that satisfies $(1/4 - \epsilon)m'$ clauses in the 3AND formula. Then the same assignment satisfies $((1/4 - \epsilon)m')^k$ clauses in the $3k$AND formula. Take these clauses as one catalog and send it to the clients corresponding to the assignment ϕ, and take their complement clauses as the other catalog and send it to the rest of the clients (which corresponds to an assignment that is the complement of ϕ). It follows that every item in each of the two catalogs is seen by $3k$ clients that are interested in it.

Consider now the case that the 3AND formula was random (hence $(1/8 + \epsilon)$ satisfiable). One of the two catalogs is received by no more than n' clients. Probabilistic analysis (omitted) shows that for this catalog it will be the case that on average, an item is received by at most $(5/2 + O(\epsilon))k$ interested clients. Hence over the two catalogs, an item is seen on average by at most $(11/4 + O(\epsilon))k$ interested clients.

This shows that one can take $\rho = 11/12$ in Theorem 6. The value of ρ can be improved by tighter analysis. □

5. EXTENSIONS

The theme of this manuscript is that there are strong connections between hardness on average and hardness of approximation. This was illustrated through the notion of R3SAT-hardness. The choice of 3SAT as our underlying assumed hard random problem is based on the fact that random instances of 3SAT have been studied extensively in the past, which helps in evaluating the plausibility of Hypothesis 2. It should be clear to the reader that this choice is to some extent arbitrary.

There are other NP-hard combinatorial optimization problems that were studied in random models. Often, spectral techniques and semidefinite programming give the strongest results known. For some problems, these results are nearly best possible. For example, for a random graph with $m > n \log n$ edges, it is possible to refute the existence of a bisection of size significantly smaller than $m/2$. For some other problems, the results are rather weak. For random graphs it is not known how to refute the existence of cliques of size somewhat smaller than $\sqrt{n}$, even though there is probably no clique of size larger than $2 \log n$. It is possible to replace Hypothesis 2 by an hypothesis that assumes hardness of average for some NP-hard problem other than 3SAT (of course, one should choose a hypothesis that is consistent with our current knowledge), and derive theorems similar in spirit to Theorem 1. We give a few examples of this.

Recall that for the dense k-subgraph problem, we proved R3SAT-hardness of approximation within some ratio $\rho < 1$. Consider the hypothesis that for random graphs, there is no polynomial time algorithm that refutes the existence of cliques of size n^ϵ. (Note however that an $n^{O(\log n)}$-time algorithm would show that there is no clique of size more than $2 \log n$.) With high probability, every subgraph with $n' = n^\epsilon$ vertices has average degree roughly $n'/2$. Hence if we could approximate dense k-subgraph within a factor better than 1/2, we could refute the above hypothesis.

Another example shows a possible distinction between Hypothesis 1 and Hypothesis 2. Consider the problem of vertex cover in 3-uniform hypergraphs (find the smallest set of vertices that touch all hyperegdes, where each hyperedge contains three vertices). Hypothesis 1 implies that this problem is hard to approximate within a factor better than 2. (Make each literal a vertex, and each clause a hyperedge containing the vertices that correspond to its three literals. A satisfying assignment gives a vertex cover with half the vertices. On the other hand, if the formula is random, then the 3-uniform hypergraph that results from this reduction is also random, and almost all vertices are needed to cover all hyperedges.) It is not known to the author whether a similar hardness result can be proved using Hypothesis 2.

As our last example, we consider a problem that was not studied as much as 3SAT or max clique. The following is proved in [20]:

THEOREM 7. *For every k, the Max kCSP problem (constraint satisfaction problems where each constrain involves k variables) is NP-hard to approximate to within a ratio of $2^{-k+O(\sqrt{k})}$. In particular, this applies to Max kAND.*

The author is not aware of any previous studies of the average case complexity of approximating Max kAND. Nevertheless, consider the following hypothesis.

HYPOTHESIS 3. *For some constant $c > 0$, for every k, for Δ a sufficiently large constant independent of n, there is no polynomial time algorithm that on most kAND formulas with n variables and $m = \Delta n$ clauses outputs* typical*, but never outputs* typical *on kAND formulas with $m/2^{c\sqrt{k}}$ satisfiable clauses. (A clause is satisfied if all its k literals are set to* true.*)*

Hypothesis 3 implies that the 2-catalog problem cannot be approximated within ratios better than 2. This can be shown by the following reduction. Create a bipartite graph in which the left hand side vertices are the $2n$ literals, the right hand side are the m clauses and their complements (where the complement of a clause is obtained by complementing every literal in it). A literal is joined by edges to all clauses that contain it. Let the size of each catalog be $r = m/2^{c\sqrt{k}}$.

If there is an assignment ψ that satisfies $m/2^{c\sqrt{k}}$ clauses of the kAND formula, then take these clauses as one catalog and send it to the literals that are set to *true* by ψ, and take the complements of these clauses as the other catalog and send it to the other literals. Each catalog item is viewed by k clients that are interested in it.

On the other hand, if the kAND formula is random, then a probabilitic argument shows that when Δ is large enough, for every set of αn literals with $0 < \alpha < 2$, for every set of $r = m/2^{\sqrt{k}}$ clauses, on average a clause from this set is connected only to $\alpha k/2 + o(k)$ of these literals. Hence any choice of two catalogs of size r contains roughly half the edges compared to the case where an exceptionally good assignment ψ exists.

6. CONCLUSIONS

The hope of the author is that this manuscript will help in establishing research directions that investigate the relations between average case complexity and approximation complexity. Much work is left for the future.

1. Try to refute hypotheses 1 and 2. Perhaps they can be refuted by reducing them to approximation problems for which we have good approximation algorithms?

2. Try to prove NP-hardness of approximation results for all or some of the problems in Theorem 1.

3. Try to derive interesting hardness of approximation results starting from problems that are average case complete in the sense of [18].

4. Try to establish a reverse connection – that hardness of approximation (e.g., of min bisection) implies hardness on average (e.g., of random 3SAT). This is not entirely hopeless, in light of Ajtai's result of this nature for lattice problems [1].

5. Significantly improve the R3SAT-hardness of approximation ratios for the problems in Theorem 1. (For example, match the hardness of approximation ratios given in Section 5, without intruducing any hypothesis beyond Hypothesis 2.)

6. Uncover more relations between average case complexity and open questions in approximation of combinatorial optimization problems.

Acknowledgements

I would like to thank Iris Dahan and Alon Rosen for their help in exploring the use of semidefinite programming for determining unsatisfiability of random 3CNF formulas.

7. REFERENCES

[1] M. Ajtai. "Generating hard instances of lattice problems". In *Proceedengs of 28th STOC*, 1996, 99–108.

[2] N. Alon, U. Feige, A. Widgerson, D. Zuckerman. "Derandomized graph products". *Computational Complexity* 5 (1995) 60–75.

[3] Paul Beame, Richard Karp, Toni Pitassi, Mike Saks. "On the complexity of unsatisfiability proofs for random k-CNF formulas". In *Proc. 30th STOC*, 1998, 561–571.

[4] Eli Ben-Sasson, Avi Wigderson. "Short proofs are narrow – resolution made simple". In *Proc. 31st STOC*, 1999, 517–526.

[5] V. Chvatal, E. Szemeredi. "Many hard examples for resolution". *JACM*, 35(4), 1988, 759–768.

[6] Y. Dodis, V. Guruswami, S. Khanna. "The 2-catalog segmentation problem". In *Proc. of SODA 1999*, 897–898.

[7] Uriel Feige, Guy Kortsarz, and David Peleg. "The dense k-subgraph problem". *Algorithmica* (2001) 29: 410–421.

[8] U. Feige and R. Krauthgamer. "A polylogarithmic approximation of the minimum bisection". *Proc. of 41st FOCS* 2000, 105–115.

[9] Uriel Feige and Michael Langberg. "Approximation algorithms for maximization problems arising in graph partitioning". *Journal of Algorithms* 41, 174–211 (2001).

[10] Ehud Friedgut. "Necessary and sufficient conditions for sharp thresholds of graph properties and the k-SAT problem". *Journal of the American Mathematical Society* 12, 1999, 1017–1054.

[11] Joel Friedman, Andreas Geordt. "Recognizing more unsatisfiable random 3SAT instances efficiently". In *Proc. 28th ICALP*, 2001, 310–321.

[12] Andreas Geordt, Michael Krivelevich. "Efficient recognition of unsatisfiable random k-SAT instances by spectral methods". In *Proc. of STACS 2001*, 294–304.

[13] M. Goemans and D. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". *JACM* 42:1115–1145, 1995.

[14] Johan Hastad. "Some optimal inapproximability results". *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, El Paso, Texas, 1997, 1–10.

[15] H. Karloff, U. Zwick. "A 7/8-approximation algorithm for MAX 3SAT?" In *Proc. 38th FOCS*, 1997, 406–415.

[16] J. Kleinberg, C. Papadimitriou, P. Raghavan. "Segmentation problems: a micro-economic view of data mining". In *Proc. 30th ACM Symposium on Theory of Computing*, 473–482, 1998.

[17] T. Leighton, S. Rao. "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms". *JACM* 46(6):787–832, 1999.

[18] L. Levin. "Average case complete problems". *SICOMP* 15, 1986, 285–286.

[19] M. Mitzenmacher. "On the hardness of finding multiple preset dictionaries". In *Proceedings of the 2001 Data Compression Conference*, 411–418.

[20] Alex Samorodnitsky, Luca Trevisan. "A PCP Characterization of NP with Optimal Amortized Query Complexity". *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, Portland, Oregon, 2000, 191–199.

[21] H. U. Simon. "On approximate solutions for combinatorial optimization problems". *SIAM J. Algebraic Discrete Methods*, 3:294–310, 1990.

[22] "Phase Transitions in Combinatorial Problems." *Theoretical Computer Science, Volume 265, Numbers 1-2.* Guest Editors: O. Dubios, R. Monasson, B. Selma, R. Zecchina. Elsevier.

[23] Uri Zwick. "Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint". In *Proc 9th SODA*, 1998, 201–210.

[24] Uri Zwick. "Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems". In *Proc. 31st STOC*, 1999, 679–687.

APPENDIX

A. A SEMIDEFINITE PROGRAM

We sketch here a semidefinite program (SDP) that gives an upper bound on the number of satisfiable clauses in a 3SAT formula. We then show that for random 3CNF formulas, this upper bound is close to m, and hence this SDP does not contradict Hypothesis 2. Moreover, even when the upper bound is specialized to 3XOR formulas (that necessarily do not have more satisfiable clauses than 3SAT formulas), the value of the upper bound is close to m. Here we present the SDP as it is applied to 3XOR formulas.

Given a max 3XOR formula ϕ with n variables and m clauses, consider the following optimization problem (that can be solved up to arbitrary precision in polynomial time, using semidefinite programming).

Fix some unit vector x_0. In inner product notation,

$$\langle x_0, x_0\rangle = 1 \tag{1}$$

With each clause i, associate four vector variables, x_i^1, x_i^2, x_i^3 and x_i^4. These vector variables correspond to the four assignments to the three literals of clause i that satisfy it. Our intention is that at most one of them will be the vector x_0 and the rest will be the 0 vector. This leads to the following constraints. For all i, j,

$$\langle x_i^j, x_i^j\rangle = \langle x_i^j, x_0\rangle \tag{2}$$

For every pair of vector variables,

$$\langle x_i^j, x_k^l\rangle \geq 0 \tag{3}$$

We require that for every two vector variables x_i^j, x_k^l that correspond to assignments that contradict each other (there is a variable that one assignment sets to true and the other to false),

$$\langle x_i^j, x_k^l\rangle = 0. \tag{4}$$

In particular, note that for $j \neq l$, $\langle x_i^j, x_i^l\rangle = 0$. Moreover, as a consequence of constraint (2) we have that for every i,

$$\sum_{j=1}^{4}\langle x_i^j, x_0\rangle \leq 1. \tag{5}$$

as we are summing the squares of the projection of x_0 on orthogonal coordinates.

The objective function is to maximize

$$\sum_{i=1}^{m}\sum_{j=1}^{4}\langle x_i^j, x_0\rangle. \tag{6}$$

Observe that the value of the objective function is at most m, by inequality 5.

The vector optimization problem gives an upper bound on the number of satisfiable clauses. This can be seen as follows. Let ψ be an assignment that satisfies m' clauses in ϕ. Then in the vector optimization problem, assign the vector x_0 to all vector variables that correspond to assignments that are consistent with ψ, and the 0 vector to all other vector variables. This gives a value of m' for the vector optimization problem.

Asymptotically, as m grows to $\Omega(n^3)$, the upper bound given by the vector optimization problem approaches $m/2$. To see this, observe that for two clauses that share the same three variables but differ in that exactly one of the variables is negated in one of the clauses and not negated in the other, the vector optimization problem can extract a value of at most 1 from the sum of the two clauses (via an argument similar to inequality (5)). When m is large enough, most clauses can be matched in such a way, giving an upper bound close to $m/2$.

As we shall now see, when m is significantly smaller than n^2, the value of the vector optimization problem is almost surely nearly m, defeating the attempt to use it in order to refute Hypothesis 2.

Given a formula ϕ with m cluases, consider a maximal subformula (subset of the clauses) ϕ' in which no two clauses share two variables. Let m' be the number of clauses in ϕ'. When $m \ll n^2$, then m' is almost m, with high probability. We now present a vector solution with value m' for ϕ'. This can easily be extended to a vector solution with value m' for ϕ, by adding 0 vectors.

The vector solution uses $n+1$ coordinates numbered from 0 to n. The vector x_0 is 1 on the 0 coordinate and 0 everywhere else. Each other vector variable x_i^j corresponds to an assignment to three literals, say the literals t, u, v. We let this vector variable have value $1/4$ on coordinate 0, and $\pm 1/4$ on each of the coordinates t, u, v, where the $\pm$ sign is assigned to agree with the polarity of the literal in the assignment. All other coordinates are 0. It can be verified that all constraints are satisfied and that the value of the objective function is m'. (The fact that no two clauses share two variables ensures that constraint (3) is satisfied.)

We remark that Guassian elimination can be used in order to check whether the 3XOR formula ϕ' is satisfiable. Hence the SDP approach misses something compared to other polynomial time algorithms.