THEORY OF COMPUTING, Volume 3 (2007), pp. 25–43
http://theoryofcomputing.org

Easily refutable subformulas of large random 3CNF formulas

Uriel Feige   Eran Ofek

Received: May 2, 2006; published: February 9, 2007.

Abstract: A simple nonconstructive argument shows that most 3CNF formulas with cn clauses (where c is a sufficiently large constant) are not satisfiable. It is an open question whether there is an efficient refutation algorithm that for most formulas with cn clauses proves that they are not satisfiable. We present a polynomial time algorithm that for most 3CNF formulas with $cn^{3/2}$ clauses (where c is a sufficiently large constant) finds a subformula with $\Theta(c^2 n)$ clauses and then uses spectral techniques to prove that this subformula is not satisfiable (and hence that the original formula is not satisfiable). Previously, it was only known how to efficiently certify the unsatisfiability of random 3CNF formulas with at least $\mathrm{poly}(\log(n)) \cdot n^{3/2}$ clauses. Our algorithm is simple enough to run in practice. We present some experimental results.

ACM Classification: F.2.2

AMS Classification: 68Q17,68Q25

Key words and phrases: proof complexity, average case analysis, Boolean formula, 3CNF, refutation, spectral methods

1 Introduction

A 3CNF formula ϕ over n variables is a set of m clauses, each one contains exactly 3 literals of three different variables. A formula ϕ is satisfiable if there exists an assignment to its n variables such that in each clause there is at least one literal whose value is true. The problem of deciding whether an input 3CNF formula ϕ is satisfiable is NP-hard, but this does not rule out the possibility of designing a good

 DOI: 10.4086/toc.2007.v003a002

heuristic for it. A heuristic for satisfiability may try to find a satisfying assignment for an input formula ϕ, in case one exists. A refutation heuristic may try to prove that no satisfying assignment exists. In this paper we present an algorithm which tries to refute an input formula ϕ. The algorithm has one sided error, in the sense that it will never say "unsatisfiable" on a satisfiable formula, but for some unsatisfiable formulas it will fail to output "unsatisfiable". It then follows that for a formula ϕ on which the algorithm outputs "unsatisfiable," its execution on ϕ is a witness for the unsatisfiability of ϕ.

How does one measure the quality of a refutation heuristic? A possible test may be to check how good the heuristic is on a random input. But then, how do we generate a random unsatisfiable formula? To answer this question we review some known properties of random 3CNF formulas. The satisfiability property has the following interesting threshold behavior. Let ϕ be a random 3CNF formula with n variables and cn clauses (each new clause is chosen independently and uniformly from the set of all possible clauses). As the parameter c is increased, it becomes less likely that ϕ is satisfiable, as there are more constraints to satisfy. In [8] it is shown that there exists c_n such that for $c < c_n(1-\varepsilon)$ almost surely ϕ is satisfiable, and for $c > c_n(1+\varepsilon)$, ϕ is almost surely unsatisfiable (for some ε which tends to zero as n increases). It is also known that $3.52 < c_n < 4.596$ [14, 12, 13]. We will use random formulas with cn clauses (for $c > c_n(1+\varepsilon)$) to measure the performance of a refutation heuristic. Specifically, the refutation heuristic is considered good if for some $c > (1+\varepsilon)c_n$ it almost surely proves that a random formula with cn clauses is unsatisfiable.

Notice that for any n, as c is increased (for $c > c_n(1+\varepsilon)$), the algorithmic problem of refutation becomes less difficult since we can always ignore a fixed fraction of the clauses. The following question is still open: how small can c be so that there is still a polynomial time algorithm which almost surely refutes random 3CNF formulas with cn clauses (c may also be an increasing function of n).

A possible approach for refuting a formula ϕ is to find a resolution proof for the unsatisfiability of ϕ. In this approach one derives new clauses implied by ϕ by combining pairs of clauses in which one clause contains a variable and the other clause contains the negation of this variable. Any satisfying assignment must satisfy at least one of the remaining literals contained in the two clauses, and hence the collection of these literals is a CNF clause implied by ϕ. A sequence of iterations of this resolution step that eventually generates the empty clause is a proof that ϕ is not satisfiable. Chvátal and Szemerédi [3] proved that a resolution proof of a random 3CNF formula with linear number of clauses is almost surely of exponential size. A result of a similar flavor for denser formulas was given by Ben-Sasson and Wigderson [2] who showed that a random formula with $n^{3/2-\varepsilon}$ clauses almost surely requires a resolution proof of size $2^{\Omega(n^{\varepsilon/(1-\varepsilon)})}$. These lower bounds imply that finding a resolution proof for a random formula is computationally inefficient.

A simple refutation algorithm can be used to refute random formula ϕ with n^2 clauses. This is done by considering only those clauses that contain a particular variable x. Fixing x to be true leaves about half of the selected clauses as a random 2CNF formula with roughly $3n/2$ clauses. A 2CNF formula with this number of clauses is almost surely not satisfiable. Moreover, any polynomial time algorithm for 2SAT can be used to certify that this particular sub-formula is not satisfiable. The same can be done when fixing x to be false, implying that no matter how x is assigned, the formula ϕ cannot be satisfied.

A new approach, introduced by Goerdt and Krivelevich in [10], gave a significant improvement and reduced the bound to $\log^7 n \cdot n^k$ clauses for efficient refutation of $2k$CNF formulas. This approach was later extended in [9] and [11] to handle also random 3CNF formulas with $n^{3/2+\varepsilon}$ and $\mathrm{poly}(\log n) \cdot n^{3/2}$

clauses, respectively. In [5, 7] it is shown how to efficiently refute a random $2k$CNF instances with at least cn^k clauses.

The difficulty of finding refutation algorithms for formulas with linearly many clauses may lead one to assume that no such algorithm exists. It is shown in [6] that this assumption (that there is no polynomial time refutation heuristic that works for most 3CNF formulas with cn clauses, where c is an arbitrarily large constant) implies that certain combinatorial optimization problems (like minimum graph bisection, the 2-catalog segmentation problem, and others) have no polynomial time approximation schemes. It is an open question whether it is NP-hard to approximate these problems arbitrarily well, though further evidence that these problems are indeed hard to approximate is given in [15].

Our refutation algorithm is based on techniques similar to those used in earlier work, such as [9, 5, 6]. We use these techniques in a different way, resulting in an algorithm that is easier to implement, easier to analyze, and works at lower densities than previous algorithms. For example, both the algorithms in [9, 11] and our algorithm perform eigenvalue computations on some random matrices derived from the random input formula ϕ. However, our matrices are much smaller (of order n rather than n^2), making the computational task easier. Moreover, the structure of our matrices is simpler, making the analysis of our algorithm simpler, and easier to apply also to formulas with fewer clauses than those in [9, 11]. As a result of this simplicity, we can show that our algorithm refutes formulas with $cn^{3/2}$ clauses, whereas the algorithms given in [9] and [11] are claimed only to refute formulas with $\Omega(n^{3/2+\varepsilon})$ and $\Omega(\mathrm{poly}(\log n) \cdot n^{3/2})$ clauses, respectively. An implementation of our algorithm refuted a random formula with $n = 50000$ variables and $27335932 < 3n^{3/2}$ clauses (see details in Section 5).

In some other respects, our algorithm is more limited then the algorithms in [9, 11]. An algorithm is said to provide *strong refutation* if it shows not only that the input 3CNF formula is not satisfiable, but also that every assignment to the variables fails to satisfy a constant fraction of the clauses. Our refutation algorithm does not provide a strong refutation. The problem of strong refutation was addressed in [4], where it was shown that variations of the algorithms of [9, 11] can strongly refute random 3CNF formulas with at least $\log^6(n) \cdot n^{3/2}$ clauses. The ability to perform strong refutation is an important issue, and its relation to approximability is discussed in [6].

2 Preliminaries

2.1 The random model

Definition 2.1. A clause is a 3-tuple of literals that belong to three different variables. The set C_n is the set of all possible clauses over n fixed variables (there are $2n \cdot 2(n-1) \cdot 2(n-2)$ such clauses).

We use the following model for generating the random formula ϕ.

Definition 2.2. A 3CNF formula ϕ is generated by choosing m clauses from C_n independently at random with repetitions. Such a random formula is denoted by $\phi \in_R C_n^m$.

Although we concentrate on a specific random model for generating random formulas, our algorithm succeeds also on other related random models. For example the formula can be a random set of $cn^{3/2}$ distinct clauses.

2.2 Notation

We write $a(n) \sim b(n)$ if $\lim_{n\to\infty} \frac{a(n)}{b(n)} \to 1$. We use the term w.h.p. (with high probability) to denote a sequence of probabilities that converges to 1 as n increases.

2.3 Efficient certification of a property

An important concept which will be frequently used is the concept of *efficient certification*. Let P be some property of graphs/formulas or any other combinatorial object. An algorithm A *certifies* the property P if the following holds:

1. On any input instance ϕ the algorithm returns either "has P" or "don't know."

2. *Soundness:* The algorithm never outputs "has P" on an instance ϕ which does not have the property P. The algorithm may output "don't know" on an instance ϕ which has the property P (the algorithm has one sided error).

We will use certification algorithms on random instances of formulas/graphs taken from some probability space $\mathcal{C}$. We shall consider properties that are almost surely true for the random object taken from $\mathcal{C}$. A certification algorithm is *complete* with respect to the probability space $\mathcal{C}$ and a property P if it almost surely outputs "has P" on an input taken from $\mathcal{C}$.

The computationally heavy part of our algorithm is certifying that two different graphs derived from the random formula ϕ do not have a large cut. One of these two graphs is random, and the other is a multigraph that is the union of 6 graphs, where each of these graphs by itself is essentially random, but there are correlations among the graphs. A cut in a graph is a partition of its vertices into two sets. The size of the cut is the number of edges with one endpoint in each part. A certification algorithm for verifying that an input graph with m edges has no cut significantly larger than $m/2$ is implicit in [16]. This algorithm is based on semi-definite programming; if the maximum cut in the input graph is of size at most $m(1/2+\varepsilon)$, then the algorithm outputs a certificate that the maximum cut in G is bounded by $m(1/2+\delta(\varepsilon))$, where $\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$. A computationally simpler algorithm can be applied if the graph is random. In [7] it is shown how to certify that in a random graph taken from $G_{n,d/n}$ the size of the maximum cut is bounded by $dn\big(1/4+\theta(1/\sqrt{d})\big)$, thus bounding the maximum cut by $m(1/2+\varepsilon)$ when d is large enough. This is done by removing the vertices of highest degree from G, and then computing the most negative eigenvalue of the adjacency matrix of the resulting graph.

2.4 An overview of our refutation algorithm

Our algorithm builds on ideas taken from earlier work ([10, 9, 6, 7, 5, 11]). This section gives an informal overview of the algorithm at a fairly detailed level. Other sections of this manuscript fill in the formal details.

The input is an arbitrary 3CNF formula ϕ with n variables and $m = cn^{3/2}$ clauses, where c is a large enough constant. Below we describe the expected behavior of our algorithm when the input formula is random. The algorithm first greedily extracts from ϕ a subformula ϕ'. This is done as follows. We say that two clauses *match* if they differ in their first literal, but agree on their second literal and on their third literal. For example, the clauses $(x_1 \vee \bar{x}_2 \vee x_3)$ and $(x_4 \vee \bar{x}_2 \vee x_3)$ match. The subformula ϕ' is constructed by greedily putting into ϕ' pairs of clauses that match, until no further matches are found in ϕ (we allow each clause of ϕ to participate in at most one matched pair of ϕ'). Let m' be the number of clauses in ϕ'. A simple probabilistic argument shows that we can expect $m' = \Theta(m^2/n^2) = \Theta(c^2 n)$. Moreover, ϕ' is essentially a union of two random (but correlated) formulas ϕ_1 and ϕ_2 (each containing one clause from every pair of clauses that are matched in ϕ'). Our algorithm will now ignore the rest of ϕ, and refute ϕ'. As explained, ϕ' is a union of two random formulas. Here we use an observation that is made in [6], that we shall call the 3XOR principle.

The 3XOR principle: In order to show that a random 3CNF formula is not satisfiable, it suffices to strongly refute it as a 3XOR formula.

Let us explain the terms used in the 3XOR principle. A clause in a 3XOR formula is satisfied if either one or three of its literals are satisfied. A strong refutation algorithm is one that shows that every assignment to the variables leaves at least a constant fraction of the clauses not satisfied (as 3XOR clauses, in our case).

A proof of the 3XOR principle is given in [6]. We sketch it here, and give it in more details in Section 3. Observe that in a random formula every literal is expected to appear the same number of times, and if the number of clauses is large enough, then things behave pretty much like their expectation. As a consequence, every assignment to the variables sets roughly half the occurrences of literals to true, and roughly half to false. Hence every assignment satisfies on average 3/2 literals per clause. Moreover, this property is easily certifiable, by summing up the number of occurrences of the n most popular literals.

Given that every assignment satisfies on average $3/2$ literals per clause, let us consider properties of satisfying assignments (if such assignments exist). The good option is that they satisfy one literal in roughly $3/4$ of the clauses, three literals in roughly $1/4$ of the clauses, and 2 literals in a negligible fraction of the clauses. This keeps the average roughly at $3/2$, and indeed nearly satisfies the formula also as a 3XOR formula, as postulated by the 3XOR principle. The bad option (which also keeps the average at $3/2$) is that the fraction of clauses that are satisfied three times drops significantly below $1/4$, implying that significantly more than $3/4$ of the clauses are satisfied either once or twice, or in other words, as a 3NAE (3-"not all equal" SAT) formula. But here, let us combine two facts. One is that for a random large enough formula, every assignment satisfies roughly $3/4$ of the clauses as a 3NAE formula. The other (to be explained below) is that there are known efficient algorithms for certifying that no assignment satisfies more than $3/4 + \varepsilon$ fraction of the clauses of a 3NAE formula. Hence for a random 3CNF formula, one can efficiently certify that the bad option mentioned above does not occur.

Having established the 3XOR principle, the next step of our algorithm makes one round of Gaussian elimination. That is, under the assumption that we are looking for near satisfiability as 3XOR (which is simply a linear equation modulo 2), we can add clauses modulo 2. Adding (modulo 2) two matched clauses, the common literals drop out, and we get a clause with only two literals whose XOR is expected to be 0, namely, a 2EQ clause (EQ for equality). For example, from the clauses $(x_1 \oplus \bar{x}_2 \oplus x_3)$ and $(x_4 \oplus \bar{x}_2 \oplus x_3)$ one gets the clause $(x_1 = x_4)$. Doing this for all pairs of matched clauses in ϕ', we get a random 2EQ formula ϕ_{2eq}. If ϕ' was nearly satisfiable as 3XOR, then ϕ_{2eq} must be nearly satisfiable as 2EQ. But if ϕ' is random, then ϕ_{2eq} is essentially a random 2EQ formula. For such formulas, every assignment satisfies roughly half the clauses. Moreover, there are known algorithms that certify this (to be explained shortly). Hence we can strongly refute ϕ_{2eq} as 2EQ, implying strong refutation of ϕ' as

3XOR, implying strong refutation of ϕ' as 3SAT, implying refutation (though not strong refutation) of ϕ as 3SAT.

Let us briefly explain here the major part that we skipped over in the description of our algorithm, namely, how to certify that a random 2EQ formula is not $1/2+\varepsilon$ satisfiable, and how to certify that a random 3NAE formula is not $3/4+\varepsilon$ satisfiable. In both cases, we reduce the certification problem to certifying that certain random graphs do not have large cuts, and then use the certification algorithms mentioned in Section 2.3. (The principle of refuting random formulas by reduction to random graph problems was introduced in [10].)

To strongly refute random 2EQ formulas, we negate the first literal in every clause, getting a 2XOR formula. Now we construct a graph whose vertices are the literals, and whose edges are the clauses. A nearly satisfying 2XOR assignment naturally partitions the vertices into two sides (those literals set to true by the assignment versus those that are set to false), giving a cut containing nearly all the edges. On the other hand, if the original 2EQ formula was random, then so is the graph, and it does not have any large cut. As explained in Section 2.3, we can efficiently certify that the graph does not have a large cut, thus strongly refuting the 2XOR formula, and hence also strongly refuting the original 2EQ formula.

To strongly refute random 3NAE formulas, we again consider a max-cut problem on a graph (in fact, a multigraph, as there may be parallel edges) whose vertices are the literals. From each 3NAE clause we derive three edges, one for every pair of literals. For example, from the 3NAE clause $(x_1,\bar{x}_2,x_3)$ we get the edges $(x_1,\bar{x}_2)$, $(\bar{x}_2,x_3)$ and (x_3,x_1). It is not hard to see that if a $3/4+\varepsilon$ fraction of the 3NAE clauses are satisfied as 3NAE, than a $\frac{2}{3}(\frac{3}{4}+\varepsilon)=\frac{1}{2}+\frac{2\varepsilon}{3}$ fraction of the edges of the graph are cut by the partition induced by the corresponding assignment. Note that in our case (of ϕ' that is the union of random ϕ_1 and random ϕ_2) this graph is essentially a union of 6 random graphs: 3 graphs derived from the clauses of ϕ_1 (one with edges derived from the first two literals in every clause, one with edges derived from the last two literals, and one from the first and third literal), and 3 graphs derived from ϕ_2. Hence it is not expected to have a cut containing significantly more than half the edges. One may certify that this is indeed the case either by using the algorithm of [16] on the whole graph, or by using the algorithm of [7] on each of the 6 random graphs separately.

Summarizing, our refutation algorithm extracts from ϕ a subformula ϕ' (composed of matched pairs of clauses), checks that in ϕ' almost all literals appear roughly the same number of times, derives from ϕ' certain graphs on $2n$ vertices and certifies that they do not have large cuts (e. g., by computing the most negative eigenvalue of their adjacency matrices). The combination of all this evidence forms a proof that ϕ is not satisfiable. If ϕ is random and large enough ($cn^{3/2}$ clauses), then almost surely the algorithm will indeed manage to collect all the desired evidence.

3 The refutation algorithm

The input formula ϕ is taken from $C_n^{cn^{3/2}}$. We will use $(?,w,\ell)$ to denote a clause in which the second and the third literals are w and ℓ, respectively, and the first literal can be any literal. The following algorithm is used to extract ϕ' from ϕ.

Algorithm Extract(ϕ)

Set $\phi_1 = \phi_2 = \emptyset$.

For every ordered pair of literals (w,ℓ):

1. Count the number of clauses in ϕ of the form $(?,w,\ell)$ and store it in $N_{(w,\ell)}$.
2. If $N_{(w,\ell)} \geq 2$ add to ϕ_1 the first appearance of a $(?,w,\ell)$ clause and add to ϕ_2 the second appearance of a $(?,w,\ell)$ clause.

Return $\phi' = (\phi_1,\phi_2)$.

Each of ϕ_1,ϕ_2 is a random formula, though clauses in ϕ_1 (and ϕ_2) are not completely independent of each other: if a clause (x,y,z) appears, then the clause (t,y,z) cannot appear. From now on we will concentrate on refuting ϕ', ignoring the rest of ϕ. The number of matched pairs in ϕ' is denoted by m (in Section 2.4 we used m to denote the number of clauses in ϕ; from here on, m will denote the number of matched pairs in ϕ').

Lemma 3.1. *Let ϕ' be the formula returned by Extract(ϕ), where $\phi \in_R C_n^{cn^{3/2}}$. W.h.p. the number of matched pairs in ϕ' is $\sim \frac{c^2 n}{8}$.*

The proof of Lemma 3.1 is deferred to Section 4. Before specifying the algorithm, we introduce additional notation and definitions which will ease the description of the algorithm.

Definition 3.2. Let ϕ be any 3CNF formula over n variables. The *graph induced by* ϕ has $2n$ vertices (corresponding to all possible literals) and the following edges. Each clause of ϕ induces three edges by taking all (unordered) pairs of literals from it (e. g., the clause $(x,y,\bar{z})$ induces the edges $(x,y),(x,\bar{z}),(y,\bar{z})$). We denote the (multi) graph induced by ϕ by G_ϕ.

Definition 3.3. Let $\phi' = (\phi_1,\phi_2)$ be a 3CNF formula with m pairs of matched clauses. The graph $G_{\phi'}$ is the graph induced by ϕ' (as in Definition 3.2). The graph $G^{2eq}_{\phi'}$ is a graph with $2n$ vertices (corresponding to all literals). Its edges are as follows: each matched pair from ϕ_1,ϕ_2, say $(x,w,\ell),(y,w,\ell)$, induces exactly one edge $(\bar{x},y)$. Note that we negate the literal which corresponds to the clause of ϕ_1.

Definition 3.4. Let ϕ be a formula with n variables and m clauses. The *imbalance* of a variable i (denoted by Im_i) is the difference in absolute value between the number of times it appears with positive polarity and the number of times it appears with negative polarity. The *total imbalance* of ϕ is $\sum_{i=1}^n Im_i$ and the *normalized imbalance* of ϕ is $(1/3m)\sum_{i=1}^n Im_i$.

If the normalized imbalance of ϕ is bounded by δ, then ϕ is δ-*balanced*.

Definition 3.5. A 3CNF formula ϕ has the $(1-\varepsilon)$ 3XOR property if for every assignment A, if A satisfies ϕ as 3CNF, then at least $1-\varepsilon$ fraction of the clauses are satisfied as 3XOR.

Definition 3.6. A graph is said to have a δ-cut if there is a partition of its vertices into two disjoint sets such that at least a δ-fraction of the edges cross this partition.

Algorithm Refute(ϕ')

1. Certify that ϕ' has the $(1-\gamma)$ 3XOR property. Specifically:
   (a) Find the normalized imbalance of ϕ' and denote it by δ'.
   (b) Certify that $G_{\phi'}$ has no $(\frac{1}{2}+\varepsilon')$-cut ($\varepsilon'$ is returned by a subroutine).
   Set $\gamma = \frac{3}{2}(\delta' + 2\varepsilon')$.
2. Certify that $G^{2eq}_{\phi'}$ has no $(\frac{1}{2}+\varepsilon)$-cut ($\varepsilon$ is returned by a subroutine).
3. If $\varepsilon + 2\gamma < \frac{1}{2}$ return "unsatisfiable," otherwise return "don't know."

In steps 1(b) and 2 of Refute(ϕ') we use as a blackbox a subroutine for bounding the maximum cuts in $G_{\phi'}$ and $G^{2eq}_{\phi'}$. This subroutine is explained in Theorem 3.10.

We first show that Refute(ϕ') is sound, i. e., whenever it returns "unsatisfiable" it holds that ϕ' can not be satisfied. This follows from Lemma 3.8 and Theorem 3.7.

Theorem 3.7 (Soundness). *Let $\phi' = (\phi_1, \phi_2)$ be a 3CNF formula composed of pairs of matched clauses. Denote by $G^{2eq}_{\phi'}$ the graph induced by ϕ' as described in Definition 3.3. The formula ϕ' is not satisfiable if all the following conditions hold:*

1. *ϕ' has the $(1-\gamma)$3XOR property,*
2. *$G^{2eq}_{\phi'}$ has no $(\frac{1}{2}+\varepsilon)$-cut,*
3. *$\varepsilon + 2\gamma < \frac{1}{2}$.*

Proof. We shall show that if ϕ' is satisfiable and has the $(1-\gamma)$3XOR property, then $G^{2eq}_{\phi'}$ has a $(1-2\gamma)$-cut. Combined with the fact that $G^{2eq}_{\phi'}$ has no $1/2+\varepsilon$ cut we derive a contradiction (since $\varepsilon+2\gamma<1/2$). Consider the cut induced on the vertices of $G^{2eq}_{\phi'}$ by a satisfying assignment A (where in one side there are all the literals whose value is true and in the other side there are all the literals whose value is false). $A(x)$ denotes the value of the literal x induced by the assignment A. By the 3XOR property of ϕ', all but γ fraction of the clauses of ϕ' are satisfied as 3XOR clauses. Hence at least a $(1-2\gamma)$ fraction of the pairs of matched clauses have both clauses in the pair satisfied by A as 3XOR. Let $(x,w,\ell),(y,w,\ell)$ be a pair such that both (x,w,ℓ) and (y,w,ℓ) are satisfied as 3XOR. It holds that: $A(x)+A(y)+2(A(w)+A(\ell)) = 0 \bmod 2$. Thus exactly one of the literals $\bar{x}, y$ is true and the other is false (under the assignment A), and the edge $(\bar{x}, y)$ induced by this pair of clauses crosses the cut. It then follows that at least $1-2\gamma$ fraction of the edges in $G^{2eq}_{\phi'}$ are cut edges. ☐

Lemma 3.8 (The 3XOR lemma). *Let ϕ be a 3CNF formula. If the following hold:*

1. *ϕ is δ-balanced, and*
2. *the graph (induced by ϕ) G_ϕ has no $(\frac{1}{2}+\varepsilon)$-cut,*

then ϕ has the $(1-\frac{3}{2}(\delta+2\varepsilon))$3XOR property.

Proof. The proof of this lemma appears in [6]; also a similar version of this lemma (for denser random 2kCNF formulas) appears in [5]. We repeat the proof for the sake of self-containment.

Denote by m the number of clauses in ϕ and let A be a satisfying assignment. We bound from above the number of satisfied appearances of literals. The assignment which maximizes the number of satisfied appearances of literals is the 'majority vote': a variable x is assigned 'true' iff it appears more times with positive polarity than with negative polarity. Using this assignment the total number of satisfied appearances is $\frac{1}{2}\left(3m+\sum_{i=1}^{n} Im_i\right)$ (where Im_i denotes the imbalance of variable i). It follows that on average each clause is satisfied at most $\frac{3}{2}+\frac{1}{2m}\sum_{i=1}^{n} Im_i=\frac{3}{2}(1+\delta)$ times.

A clause is satisfied as 3AND by A if all its literal are satisfied by A. We next show that the fraction of clauses satisfied by A as 3AND is at least $1/4-3\varepsilon/2$. Equivalently it is enough to show that the fraction of clauses satisfied as 3NAE, denoted by β, is at most $3/4+3\varepsilon/2$ (since A is a satisfying assignment). Consider the graph G_ϕ induced by ϕ. We remind the reader that each clause of ϕ contributes a "triangle" of 3 edges to G_ϕ (e. g., the clause $(x,\bar{y},z)$ contributes the edges $(x,\bar{y}),(\bar{y},z),(x,z)$). Consider the cut induced by the satisfying assignment A. Each clause satisfied as 3NAE contributes exactly 2 edges to the cut, thus this cut has at least $2\beta m$ edges. But G_ϕ has no $(1/2+\varepsilon)$-cut. It follows that $2\beta m \leq (1/2+\varepsilon)3m$, giving $\beta \leq 3/4+3\varepsilon/2$ as needed.

It remains to show that all but a small fraction of the clauses are satisfied as 3XOR by A. Denote by $\alpha_1,\alpha_2,\alpha_3$ the fraction of clauses which are satisfied exactly once, exactly twice and exactly 3 times respectively ($\sum_{i=1}^{3}\alpha_i=1$). We already know that $\alpha_3 \geq 1/4-3\varepsilon/2$ and that each clause is satisfied at most $3/2+\delta$ times on average, thus

$$\frac{3(1+\delta)}{2} \geq 3\cdot\alpha_3+2\cdot\alpha_2+1\cdot\alpha_1 \ .$$

Substituting $\alpha_1=(1-\alpha_3-\alpha 2)$ and α_3 with $\frac{1}{4}-\frac{3}{2}\varepsilon$ yields that $\alpha_2 \leq \frac{3}{2}(\delta+2\varepsilon)$. ☐

The following two Theorems show that our refutation algorithm refutes most formulas $\phi \in_R C_n^{cn^{3/2}}$.

Theorem 3.9. *For any $\varepsilon>0$ there is a constant $c=c(\varepsilon)$ such that w.h.p. over the choice of $\phi \in_R C_n^{cn^{3/2}}$:*

(a) the subformula ϕ' is ε-balanced, and

(b) each of the graphs $G_{\phi'}$ and $G_{\phi'}^{2eq}$ has no $(\frac{1}{2}+\varepsilon)$-cut.

It is well known and follows from standard probabilistic calculations that a random graph (with large enough average degree) has no $(1/2+\varepsilon)$-cut, and that a random formula is ε-balanced. The distributions of $\phi',G_{\phi'}$ are close enough to the standard models of random formulas/graphs respectively, so that the proof techniques used for the random cases can be applied also in our case. The proof of Theorem 3.9 is deferred to Section 4.2.

Theorem 3.10. *There is a polynomial time algorithm that finds the imbalance of a 3CNF formula. There is a polynomial time algorithm that for every graph G that has no $(\frac{1}{2}+\varepsilon)$-cut, certifies that G has no $(\frac{1}{2}+\delta)$-cut, where $\delta(\varepsilon)\to 0$ as $\varepsilon\to 0$.*

Proof. Finding the normalized imbalance of a formula ϕ is done by counting positive and negative appearances for every variable, computing its imbalance and averaging.

An algorithm for certifying a bound on the maximum cut of a graph is given in [16]. Given a graph with m edges whose maximum cut is bounded by $m(1/2+\varepsilon)$ this algorithm produces a proof that the input graph has no cut of cardinality $m(1/2+\delta)$. This algorithm has the property that $\delta \to 0$ as $\varepsilon \to 0$. □

Corollary 3.11 (Almost-completeness). *For sufficiently large constant c, the refutation algorithm w.h.p. returns "unsatisfiable" for a random formula $\phi \in_R C_n^{cn^{3/2}}$.*

Proof. Using Theorems 3.9, 3.10 we will show that by taking c to be a sufficiently large constant, the terms ε, γ from Refute(ϕ') can be made arbitrary small (and thus $\varepsilon + 2\gamma < 1/2$). The term γ equals $(3/2)(\delta' + 2\varepsilon')$ where δ' is the imbalance of ϕ' and $(1/2+\varepsilon')$ is the bound returned by the algorithm of [16] when applied to $G_{\phi'}$. By Theorem 3.9, $\delta' \to 0$ as c increases. Furthermore, the value of the maximum cut in both $G_{\phi'}, G_{\phi'}^{2eq}$ approaches $1/2$ as c increases. It then follows, using Theorem 3.10, that the bounds $1/2+\varepsilon'$ and $1/2+\varepsilon$ returned by the certification algorithm (of [16]) applied to $G_{\phi'}$ and $G_{\phi'}^{2eq}$, respectively, can be made arbitrary close to $1/2$ by setting c to be a sufficiently large constant.

□

4 Proofs of Lemma 3.1 and Theorem 3.9

4.1 Proof of Lemma 3.1

Proof of Lemma 3.1. For two random clauses the probability that they induce a matched pair of clauses is

$$p \triangleq \frac{1}{2n}\frac{1}{2n-2} \sim \frac{1}{4n^2} \quad .$$

Thus, the expected number of pairs that match is

$$\mu = \binom{cn^{3/2}}{2} p \sim \frac{c^2n^3}{2}\frac{1}{4n^2} \sim \frac{c^2n}{8} \quad . \tag{4.1}$$

Let X denote the number of pairs of clauses in ϕ that match. We use the second moment to show that w.h.p. $X \sim \mu$. By Chebyshev's inequality

$$\Pr[|X-\mu| > \varepsilon\mu] < \frac{\mathrm{Var}(X)}{\varepsilon^2\mu^2} \quad . \tag{4.2}$$

For every two clause locations in ϕ (e.g., first clause and third clause) we set an indicator random variable X_i $(i = 1,2,\ldots,\binom{cn^{3/2}}{2})$ to be 1 if the respective two clauses match and otherwise 0. For $i \neq j$ we say that $i \sim j$ if the pair i and pair j share one clause location, and otherwise $i \nsim j$. (Note that if pair i and pair j share two clause locations, then $i = j$. Note also that $i \nsim j$ might share the same clause without sharing a clause location, if a certain clause happened to appear twice in ϕ.) For any fixed i we let

$$\Delta^* = \sum_{j:\ j\sim i} \Pr[X_j \mid X_i] \quad . \tag{4.3}$$

¿From symmetry, Δ^* does not depend on i.

$$\begin{aligned}
\mathrm{Var}(X) &= \mathbb{E}\left[X^2\right] - \mu^2 = \mathbb{E}\big[(\sum_i X_i)^2\big] - \mu^2 = \mathbb{E}\Big[\sum_i X_i^2 + \sum_{i\neq j} X_i X_j\Big] - \mu^2 \\
&= \mu - \mu^2 + \sum_{i\neq j} \Pr[X_i X_j] = \mu - \mu^2 + \sum_i \Pr[X_i]\left(\sum_{j:\ j\nsim i} \Pr[X_j] + \sum_{j:\ j\sim i} \Pr[X_j \mid X_i]\right) \\
&\leq \mu - \mu^2 + \underbrace{\sum_i \Pr[X_i] \sum_j \Pr[X_j]}_{\mu^2} + \sum_i \Pr[X_i] \underbrace{\sum_{j:\ j\sim i} \Pr[X_j \mid X_i]}_{\Delta^*} \\
&= \mu + \sum_i \Pr[X_i]\Delta^* = \mu(1+\Delta^*) \ . \qquad (4.4)
\end{aligned}$$

Substituting $\mathrm{Var}(X)$ with $\mu(1+\Delta^*)$ in inequality (4.2) we derive

$$\Pr[|X-\mu| > \varepsilon\mu] < \frac{\mu(1+\Delta^*)}{\varepsilon^2\mu^2} \leq \frac{1+\Delta^*}{\varepsilon^2\mu} \ . \qquad (4.5)$$

Thus, since $\mu = \omega(1)$, it suffices to show that $\Delta^* = o(\mu)$. It holds that

$$\Delta^* \leq 2(cn^{3/2} - 2)p = o(\mu) \ . \qquad (4.6)$$

So far we showed that w.h.p. $X \sim \mu$. Note that X may over-count the number of matched pairs in ϕ'. The reason is that in ϕ there are expected to be sets of three or more clauses in which any two clauses match. From each such set, $\mathrm{Extract}(\phi)$ takes to ϕ' only the first two clauses of the set.

For $i \geq 3$, we call a set of i clauses *bad* if each two clauses of the set match. Let Y_i be the number of bad sets of size i in ϕ. The number of matched clauses in ϕ' is at least $X - \sum_{i\geq 3}\binom{i}{2}Y_i$. Thus, in order to show that the number of matched pairs in ϕ' is $\sim \mu$ it is enough to prove that

$$\mathbb{E}\left[\sum_{i\geq 3}\binom{i}{2}Y_i\right] = o(\mu) \ .$$

Then, using Markov's inequality we derive that w.h.p. $\sum_{i\geq 3}\binom{i}{2}Y_i = o(\mu)$. It holds

$$\mathbb{E}\left[\binom{i}{2}Y_i\right] = \binom{i}{2}\binom{cn^{3/2}}{i}p^{i-1} \leq \frac{i^2}{2}\left(\frac{cn^{3/2}e}{i}\right)^i \left(\frac{1}{4n^2(1-1/n)}\right)^{i-1} \ . \qquad (4.7)$$

Thus, the sum $\sum_{i\geq 3}\mathbb{E}\big[\binom{i}{2}Y_i\big]$ is bounded by the sum of a geometric sequence whose first term ($i=3$) is $o(\mu)$. It follows that $\mathbb{E}\big[\sum_{i\geq 3}\binom{i}{2}Y_i\big] = o(\mu)$.

□

4.2 Proof of Theorem 3.9

Let $P \subset C_n \times C_n$ be the set of all possible matched pairs of clauses, and let P^m be the set of all m-tuples of matched pairs of clauses. For a pair of matched clauses $(c_1, c_2) \in P$, the *inducing pair* is the pair formed by the second and third literals in each of the two matched clauses c_1, c_2. Let m denote the number of matched pairs of clauses in ϕ'. Note that $\phi' \in P^m$, however not all the elements of P^m are in the support of ϕ'. We denote by $\binom{P}{m} \subset P^m$ the support of ϕ', i. e., the collection of all (ordered) m matched pairs for which every matched pair of clauses has a distinct inducing pair. We claim that ϕ' is a random element of $\binom{P}{m}$.

Lemma 4.1. *Given that ϕ' has m matched clauses, the set of inducing pairs is a random set of m different ordered pairs of literals (each pair has two distinct literals).*

Proof. Let C denote the set of clauses that have a matching clause (from the original formula ϕ). Denote by L the set of inducing pairs, i. e., pairs that participate as the second and third literal in one of the clauses of C. The set L may be any set of distinct ordered pairs of size m. By symmetry, any such set is equally likely to be L. The explanation is as follows. Assume we expose the indices of the clauses in C and also the partition of C into equivalent classes (each equivalent class is a maximal set of clauses that have the same second and third literals). Given this information, for each choice of L, the probability that L is the set of inducing pairs is the same (for any L the number of ways to match the pairs of L with the equivalent classes is the same; additionally, the probability for all other clauses not in C to avoid all the pairs of L in the second and third literals is the same). $\square$

Proof of Theorem 3.9. ¿From here on we will assume that m is a fixed number and that $m \sim c^2n/8$ (this is justified by Lemma 3.1). The formula ϕ' is a random element of $\binom{P}{m}$. Let $\phi'' \in_R P^m$ (i. e., ϕ'' is composed of m random and independent samples from P). Denote by Ω' the event that every matched pair of clauses in ϕ'' has a distinct inducing pair. Conditioned on Ω', ϕ'' has the distribution of ϕ'. As Lemma 4.2 shows, the event Ω' is not too rare (the proof of Lemma 4.2 is deferred to the end of this section).

Lemma 4.2. *Let $m = \frac{c^2n}{8}$. For $\phi'' \in_R P^m$ it holds that* $\Pr[\Omega'] \geq e^{-c^4/128}$

Furthermore, as Lemma 4.3 states, $G_{\phi''}$ is unlikely to have a large cut.

Lemma 4.3. *Let $m \sim \frac{c^2n}{8}$. For $\phi'' \in_R P^m$ with probability $1 - o(1)$ it holds that $G_{\phi''}$ has no $(\frac{1}{2} + \frac{4}{c})$-cut.*

Combining Lemmas 4.2, 4.3 we now show that $G_{\phi'}$ is unlikely to have a large cut. The reasoning is as follows:

$$\Pr_{\phi' \in_R \binom{P}{m}}[G_{\phi'} \text{ has } (\tfrac{1}{2} + \tfrac{4}{c}) \text{ cut}] = \Pr_{\phi'' \in_R P^m}[G_{\phi''} \text{ has } (\tfrac{1}{2} + \tfrac{4}{c}) \text{ cut} \mid \Omega'] \tag{4.8}$$

$$\leq \frac{\Pr_{\phi''}[G_{\phi''} \text{ has } (\frac{1}{2} + \frac{4}{c}) \text{ cut}]}{\Pr_{\phi''}[\Omega']} \leq \frac{o(1)}{e^{-c^2/128}} = o(1) \ , \tag{4.9}$$

where the last equality is because c is a fixed constant.

A similar argument shows that if w.h.p. $G^{2eq}_{\phi''}$ has no $(1/2+\varepsilon)$-cut and ϕ'' is ε-balanced, then w.h.p. $G^{2eq}_{\phi'}$ has no $(1/2+\varepsilon)$-cut and ϕ' is ε-balanced. Hence, we only need to prove the following lemmas.

Lemma 4.4. *Let $m \sim \frac{c^2n}{8}$. For $\phi'' \in_R P^m$ with probability $1 - o(1)$ it holds that $G^{2eq}_{\phi''}$ has no $(\frac{1}{2} + \frac{5}{c})$-cut.*

Lemma 4.5. *Let $m \sim \frac{c^2 n}{8}$. For $\phi'' \in_R P^m$, with probability $1-o(1)$ it holds that ϕ'' is $\frac{3}{c}$-balanced.*

□

To complete the proof of Theorem 3.9 we now give the proofs of Lemmas 4.3, 4.4, 4.5, and 4.2.

Proof of Lemma 4.3. Fix a partition (V_1,V_2) of the vertices of $G_{\phi''}$. We denote by $W_{\phi''}(V_1,V_2)$ the number of edges crossing the cut (V_1,V_2) in $G_{\phi''}$. For $\phi'' \in_R P^m$ the expectation of $W_{\phi''}(V_1,V_2)$ is at most $6m(1/2+1/n)$. For any formula $\phi'' \in P^m$ we let $f(\phi'')$ be equal to $W_{\phi''}(V_1,V_2)$. Let X_i be the expected value of f after exposing the first i pairs of ϕ'' (for $i=0,1,\ldots,m$). The sequence $X_0,X_1,\ldots,X_m$ is a martingale. The following two facts:

1. for any $\phi'' \in P^m$, changing one matched pair (an element of P) can change the value of f by at most 4 (each clause forms a triangle that contributes at most 2 edges to the cut), and
2. ϕ'' is taken from a product measure P^m,

imply that for every i it holds that $|X_i - X_{i+1}| \le 4$ (see Theorem 7.4.1 from [1]). Azuma's inequality implies that for any $\lambda > 0$

$$\Pr[f(\phi'') - 6m(\tfrac{1}{2}+\tfrac{1}{n}) > \lambda] < e^{-\frac{\lambda^2}{2m4^2}} \ .$$

Setting $\lambda = \frac{17m}{c}$ and using $m \sim \frac{c^2 n}{8}$ we derive

$$\Pr_{\phi'' \in_R P^m}[W_{\phi''}(V_1,V_2) > 6m(\tfrac{1}{2}+\tfrac{4}{c})] < 2^{-1.1n} \ .$$

Using the union bound over all possible cuts we derive that w.h.p. (for $\phi'' \in_R P^m$) the graph $G_{\phi''}$ has no $(\frac{1}{2}+\frac{4}{c})$-cut.

□

The proof of Lemma 4.4 is very similar to the proof of Lemma 4.3, details are omitted.

Proof of Lemma 4.5. We first bound the expected imbalance of ϕ''. The total imbalance of ϕ'' is bounded by the sum of the imbalances of ϕ_1,ϕ_2 (where ϕ_1/ϕ_2 are formed by taking the first/second clause from each matched pair of ϕ''). Since ϕ_2 has the same distribution of ϕ_1, it is enough to bound the expected total imbalance of ϕ_1 (and then multiply by two). The total imbalance of ϕ_1 is the sum of the imbalances of all variables in ϕ_1, i. e., $\sum_{i=1}^n Im_i$ (the imbalance of x_i in ϕ_1 is denoted by Im_i).

For any variable x_i we denote by d_i the number of appearances of x in ϕ_1. For $\phi'' \in_R P^m$ it holds that $\sum_{i=1}^n \mathbb{E}[d_i] = 3m$. By symmetry, for every i it holds that $\mathbb{E}[d_i] = 3m/n$. We denote $d \triangleq 3m/n$. Given that $d_i = k$ the polarities of the appearances of x_i are still random. Given that $d_i = k$, the imbalance of x_i is the absolute value of the sum of k independent random variables, where each random variable has probability $1/2$ of being 1 and probability $1/2$ of being -1. Hence $\mathbb{E}[Im_i^2 \mid d_i = k] = k$. It then follows that

$$\mathbb{E}[Im_i^2] = \sum_k \Pr[d_i = k] \cdot \mathbb{E}[Im_i^2 \mid d_i = k] = \sum_k k \Pr[d_i = k] = \mathbb{E}[d_i] = d \ , \tag{4.10}$$

where all probabilities and expectations are taken over $\phi'' \in_R P^m$. Using the convexity of the square function

$$\mathbb{E}[Im_i] \leq \sqrt{\mathbb{E}[Im_i^2]} \leq \sqrt{d} \quad . \tag{4.11}$$

So far we showed that $\mathbb{E}_{\phi'' \in_R P^m}\left[\sum_{i=1}^{n} Im_i\right] \leq n\sqrt{d}$. Thus, for $\phi'' \in_R P^m$ the total imbalance of ϕ'' is expected to be less than $2n\sqrt{d}$. We will now show that the total imbalance of ϕ'' is not likely to be too large relative to $2n\sqrt{d}$. For any $\phi'' \in P^m$ we let $f(\phi'')$ be the total imbalance of ϕ''. Changing one matched pair of clauses in ϕ'' changes the total imbalance of ϕ'' by at most 12. Azuma's inequality implies that for any $\lambda > 0$

$$\Pr_{\phi'' \in_R P^m}\left[f(\phi'') - 2n\sqrt{d} > \lambda\right] < e^{-\frac{\lambda^2}{2m12^2}} \quad .$$

Setting $\lambda = n\sqrt{d}$ and using $d = 3m/n$ we derive

$$\Pr_{\phi'' \in_R P^m}[\text{the total imbalance of } \phi'' > 3n\sqrt{d}] < e^{-\frac{n}{96}} \quad .$$

It then follows that the normalized imbalance of ϕ'' is w.h.p. bounded by $3n\sqrt{d}/6m \leq 3/c$ (using $m \sim c^2 n/8$ and $d = 3m/n$). □

Proof of Lemma 4.2. We generate ϕ'' iteratively by choosing each time (independently) a random element of P. For each new random element of P, the probability for it to have an inducing pair which is different from all previous inducing pairs is $\geq 1 - m/N$, where $N = 2n \cdot (2n-2)$ is the number of possible inducing pairs. It then follows that with probability of at least

$$\left(1 - \frac{m}{2n(2n-2)}\right)^m \overset{(1)}{\geq} \exp\left(-m\left(\frac{m}{2n^2(1-1/n)}\right)\right) \overset{(2)}{\geq} e^{-c^4/128} \quad ,$$

each of the matched pairs in ϕ'' has a distinct inducing pair. Inequality (1) is because $1 - x \geq e^{-2x}$ holds for every $x \in [0, 1/2]$. The constant in the exponent following Inequality (2) is derived by taking $m = c^2 n/8$. □

5 Practical considerations for the refutation algorithm

Recall that our refutation algorithm extracts from ϕ a subformula ϕ' that contains matched pairs of clauses, and then refutes ϕ'. The longer ϕ' is, the easier it is to refute it. For simplicity, we matched a pair of clauses only if they agreed on their last two literals. Moreover, every clause of ϕ participated in at most one pair of matched clauses in ϕ', even though a clause may be eligible to participate in more than one matched pair. In practical implementations, it is advantageous not to have these restrictions, and thus get a longer formula ϕ'. In particular, we may allow the same clause to participate in several pairs of matched clauses, by duplicating it. More importantly, we may match any two clauses that share two variables (regardless of the polarity of the variables, and of their location within the clauses). For example, the two clauses (x, w, ℓ) and $(\bar{w}, \ell, y)$ can be matched. If ϕ' is satisfied by an assignment A that has the 3XOR property, then one step of Gaussian elimination gives in this case $A(x)+A(w)+A(\ell)+A(y)+A(\bar{w})+A(\ell)=0 \bmod 2$, thus $A(x)+A(y)=1 \bmod 2$. Hence, we will associate the edge (x,y) with this pair of matched clauses so that the edge induced by this pair in $G^{2eq}_{\phi'}$ will cross the cut which corresponds to the assignment A. Using the principles above, the number of pairs of matched clauses that one expects to extract from a random formula of length $cn^{3/2}$ is roughly $\binom{3cn^{3/2}}{2}/\binom{n}{2} \simeq 9c^2n$.

We used the principles above to implement the algorithm in practice. In the current implementation, the problem of refuting ϕ' is reduced to strong refutation of two 2XOR formulas. We performed 2 eigenvalue computations on matrices of size $n\times n$, whereas the original refutation algorithm performed eigenvalue computations on matrices of size $2n\times 2n$. Our implementation uses the conditions of Theorem 5.2 to refute ϕ'. Before stating Theorem 5.2 we need the following definition.

Definition 5.1. Let ϕ be a 2XOR formula with m clauses and n variables. A_ϕ is the following $n\times n$ symmetric matrix associated with ϕ. Initially A_ϕ is the zero matrix. For each clause of the forms $(x,\bar{y})$ or $(\bar{x},y)$ we add $+1$ to positions $A(x,y)$ and $A(y,x)$. For each clause of the forms (x,y) or $(\bar{x},\bar{y})$ we add -1 to positions $A(x,y)$ and $A(y,x)$.

A similar matrix can be defined for a 2EQ formula, just by reducing the 2EQ formula into a 2XOR formula.

Theorem 5.2. *Let ϕ' be a 3CNF formula with m pairs of matched clauses and n variables. Let ϕ_{2xor} be the 2XOR formula with* $6m$ *clauses induced by replacing each 3CNF clause of ϕ' by three 2XOR clauses (one for every two literals). Let ϕ_{2eq} be the 2EQ formula with m clauses induced by adding pairs of matched clauses modulo* 2. *If the following hold then ϕ' is not satisfiable:*

(1) ϕ' is δ-balanced.

(2) Let $\lambda_{2xor},\lambda_{2eq}$ denote the largest eigenvalues of $A_{\phi_{2xor}},A_{\phi_{2eq}}$ respectively; then

$$3\delta+\frac{n}{4m}(\lambda_{2xor}+\lambda_{2eq})<\tfrac{1}{2}\ .$$

The proof of Theorem 5.2 will follow shortly.

Lemma 5.3 (2XOR strong refutation). *Let ϕ be a 2XOR formula with m clauses. If λ is the maximum eigenvalue of A_ϕ then ϕ is at most $(\frac{1}{2}+\varepsilon)$ satisfiable, for $\varepsilon=\frac{\lambda n}{4m}$.*

Proof. Let T be an assignment satisfying the most clauses as 2XOR. Let x be the ± 1 vector which corresponds to T: x_i equals $+1$ if $T(i)=$ true, otherwise $x_i=-1$. It holds that $x^tA_\phi x=A_\phi\bullet xx^t$ (where for any two matrices A,B of the same dimensions, $A\bullet B=\sum_{i,j}A(i,j)B(i,j)$). Every 2XOR clause satisfied by T contributes 2 to $A_\phi\bullet xx^t$ whereas every unsatisfied clause contributes -2. If T satisfies exactly $(1/2+\varepsilon)m$ clauses, then

$$\frac{x^tA_\phi x}{x^tx}=\frac{(\frac{1}{2}+\varepsilon)m(+2)+(\frac{1}{2}-\varepsilon)m(-2)}{n}=4\varepsilon m/n\ . \tag{5.1}$$

Since $x^tA_\phi x/x^tx\le\lambda$ we derive $\varepsilon\le\lambda n/4m$.

□

Lemma 5.4 (3XOR property certification). *Let ϕ be a 3CNF formula which is δ-balanced with m clauses and n variables. Assume that the 2XOR formula induced by replacing each 3CNF clause by 3 2XOR clauses is at most $(\frac{1}{2}+\gamma)$ satisfiable. Then ϕ has the $(1-\frac{3}{2}(\delta+2\gamma))$3XOR property.*

Proof. Assume that the assignment T satisfies ϕ as 3CNF. Denote by α_i the number of clauses satisfied exactly once, twice, three times respectively ($\sum_{i=1}^{3}\alpha_i = 1$). We need to upper bound α_2. It can not be that α_3 is too small, as in such case there are many satisfied clauses in the induced 2XOR formula: if a clause of ϕ is satisfied exactly once or exactly twice, then out of the three 2XOR clauses induced by it, two are satisfied as 2XOR. Since the number of satisfied 2XOR clauses (in the induced 2XOR formula) is bounded by $3m(1/2+\gamma)$, we derive $2m(\alpha_1+\alpha_2) \le 3m(1/2+\gamma)$, or equivalently $\alpha_3 \ge 1/4-3\gamma/2$. The imbalance of ϕ is bounded by δ, thus the number of satisfied literals in ϕ is at most $\frac{3}{2}(1+\delta)m$. This implies

$$\frac{3(1+\delta)}{2} \ge \alpha_1+2\alpha_2+3\alpha_3.$$

Using the facts: $\alpha_1 = 1-\alpha_2-\alpha_3$ and $\alpha_3 \ge \frac{1}{4}-\frac{3}{2}\gamma$ we deduce

$$\tfrac{3}{2}(1+\delta) \ge \alpha_2+1+2(\tfrac{1}{4}-\tfrac{3}{2}\gamma), \tag{5.2}$$

and thus $\frac{3}{2}(\delta+2\gamma) \ge \alpha_2$. □

Proof of Theorem 5.2. Assume that ϕ' is satisfiable as 3CNF. We show that property (1) contradicts property (2). Set

$$\varepsilon_{2xor} \triangleq \frac{\lambda_{2xor}n}{4\cdot 6m}, \qquad \text{and} \qquad \varepsilon_{2eq} \triangleq \frac{\lambda_{2eq}n}{4m}.$$

By Lemma 5.3 ϕ_{2xor} and ϕ_{2eq} are at most $(1/2+\varepsilon_{2xor})$ and $(1/2+\varepsilon_{2eq})$ satisfied, respectively . It then follows, using Lemma 5.4, that ϕ' is $(1-(3/2)(\delta+2\varepsilon_{2xor}))$ satisfied as 3XOR. Each pair of matched clauses of ϕ' for which both clauses are satisfied as 3XOR yields a satisfied 2EQ clause of ϕ_{2eq}. As ϕ_{2eq} is at most $(1/2+\varepsilon_{2eq})$ satisfied, we conclude that

$$1-3(\delta+2\varepsilon_{2xor}) \le \tfrac{1}{2}+\varepsilon_{2eq}, \quad \text{or equivalently} \tag{5.3}$$

$$3(\delta+2\varepsilon_{2xor})+\varepsilon_{2eq} \ge \tfrac{1}{2}\ . \tag{5.4}$$

Substituting ε_{2xor} and ε_{2xor} according to the definitions in (5) we derive

$$3\delta+\frac{n}{4m}(\lambda_{2xor}+\lambda_{2eq}) \ge \tfrac{1}{2}\ ,$$

which contradicts property (2). □

We generated several random formulas with $n = 5\cdot 10^4$ variables and $27335932 = \lceil 2.445\cdot n^{3/2}\rceil$ clauses. Our algorithm refuted all of them (our current implementation fails to refute formulas of significantly lower clause density). We give more detailed results for one specific (though typical) run. Our algorithm extracted a subformula ϕ' with $m = 2689832$ pairs of matched clauses. Table 1 summarizes the values computed by the algorithm along with a heuristic estimation of what we could have expected them to be.

	m	λ_{2eq}	λ_{2xor}	δ	Bound
Algorithm	2689832	20.8961	54.6503	0.048662	$0.49706 < \frac{1}{2}$
Heuristic bound:	2690112	20.7454	50.8157	0.05565	$0.49946 < \frac{1}{2}$
(using formula)	$\approx 9c^2n$	$2\sqrt{18c^2}$	$2\sqrt{108c^2}$	$\sqrt{\frac{n}{6m}}$	$\frac{n}{4m}(\lambda_{2eq}+\lambda_{2xor})+3\delta$

Table 1: Results for a random formula with $5 \cdot 10^4$ variables and 27335932 clauses.

Let us explain the heuristic bounds used in the table. To estimate the largest eigenvalue of a symmetric matrix we use the formula $2\sqrt{d}$ where d is the average ℓ_1 norm of each of the rows. This bound is known to be true for various random graph models, but apparently is too optimistic for $A_{\phi_{2xor}}$. To estimate the imbalance δ we assume that each variable appears exactly $6m/n$ times, each time with random polarity. The difference between the number of positive and negative appearances behaves like the distance from 0 when performing a random walk of length $6m/n$ on $\mathbb{Z}$ (starting from 0). The expected square of the distance is $6m/n$, and $\sqrt{6m/n}$ is an upper bound on the expected distance. We estimated the expected normalized imbalance as $n\sqrt{6m/n}/6m = \sqrt{n/6m}$.

A few words about the implementation of our algorithm. The part of extracting the subformula ϕ' was implemented in C. The other parts (computing the imbalance and the eigenvalues) were implemented in Matlab. To save memory we used Matlab's sparse matrix objects. The heavy part of the algorithm was computing the largest eigenvalues of the two matrices $A_{\phi_{2xor}}, A_{\phi_{2eq}}$. This part took 63 minutes on an Intel Xeon CPU 1700MHz with 256K cache and 2Gbyte memory (running the Linux operating system).

It may be interesting to see if other refutation algorithms (which may use various optimized versions of resolution, OBDDs, backtracking, to name a few of the common algorithmic principles used for refutation) can handle random formulas with as many variables as those handled by our algorithm. We have not made a serious attempt to check this.

Acknowledgements

This research was supported by a grant from the G.I.F., the German-Israeli Foundation for Scientific Research and Development. We thank Amin Coja-Oghlan for useful discussions.

References

[1] * N. ALON AND J. SPENCER: *The Probabilistic Method.* John Wiley and Sons, 2002. 4.2

[2] * E. BEN-SASSON AND A. WIGDERSON: Short proofs are narrow— resolution made simple. *J. ACM*, 48(2):149–169, 2001. [JACM:10.1145/375827.375835]. 1

[3] * V. CHVÁTAL AND E. SZEMERÉDI: Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988. [JACM:10.1145/48014.48016]. 1

[4] * A. COJA-OGHLAN, A. GOERDT, AND A. LANKA: Strong refutation heuristics for random k-SAT. *Combinatorics, Probability and Computing*, 16(1):5–28, 2007. Conference version appeared in Proc. 8th Internat. Workshop on Randomization and Computation (RANDOM '04), volume 3122 of LNCS, pp. 310–321. Springer, 2004. [doi:10.1017/S096354830600784X, Springer:x1tlgm3cexcj]. 1

[5] * A. COJA-OGHLAN, A. GOERDT, A. LANKA, AND F. SCHADLICH: Techniques from combinatorial approximation algorithms yield efficient algorithms for random $2k$-sat. *Theoretical Computer Science*, 329:1–45, 2004. [TCS:10.1016/j.tcs.2004.07.017]. 1, 2.4, 3

[6] * U. FEIGE: Relations between average case complexity and approximation complexity. In *Proc. 34th STOC*, pp. 534–543. ACM Press, 2002. [STOC:509907.509985]. 1, 2.4, 3

[7] * U. FEIGE AND E. OFEK: Spectral techniques applied to sparse random graphs. *Random Structures and Algorithms*, 27(2):251–275, 2005. [RSA:10.1002/rsa.20089]. 1, 2.3, 2.4

[8] * E. FRIEDGUT AND J. BOURGAIN: Sharp thresholds of graph properties, and the k-sat problem. *Journal of the American Mathematical Society*, 12(4):1017–1054, 1999. [JAMS:1999-12-04/S0894-0347-99-00305-7]. 1

[9] * J. FRIEDMAN, A. GOERDT, AND M. KRIVELEVICH: Recognizing more unsatisfiable random 3-sat instances efficiently. *SIAM J. on Computing*, 35(2):408–430, 2005. [SICOMP:10.1137/S009753970444096X]. 1, 2.4

[10] * A. GOERDT AND M. KRIVELEVICH: Efficient recognition of random unsatisfiable k-SAT instances by spectral methods. In *Proc. Ann. Symp. on Theoretical Aspects of Computer Science (STACS '01)*, volume 2010 of *LNCS*, pp. 294–304. Springer, 2001. [STACS:27cr0lrbpr7px7y3]. 1, 2.4

[11] * A. GOERDT AND A. LANKA: Recognizing more random 3-sat instances efficiently. LICS'03 Workshop on Typical Case Complexity and Phase Transitions, 2003. 1, 2.4

[12] * M. HAJIAGHAYI AND B. SORKIN: The satisfiability threshold of random 3-SAT is at least 3.52, 2003. [arXiv:math/0310193]. 1

[13] * S. JANSON, Y. STAMATIOU, AND M. VAMVAKARI: Bounding the unsatisfiability threshold of random 3-sat. *Random Structures and Algorithms*, 17(2):103–116, 2000. [RSA:10.1002/1098-2418(200009)17:2¡103::AID-RSA2¿3.0.CO;2-P]. 1

[14] * A. KAPORIS, L. KIROUSIS, AND E. LALAS: The probabilistic analysis of a greedy satisfiability algorithm. *Random Structures and Algorithms*, 28(4):444–480, 2006. [RSA:10.1002/rsa.20104]. 1

[15] * S. KHOT: Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proc. 45th FOCS*, pp. 136–145. IEEE Computer Society, 2004. [FOCS:10.1109/FOCS.2004.59]. 1

[16] * U. ZWICK: Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems. In *Proc. 31st STOC*, pp. 679–687. ACM Press, 1999. [STOC:301250.301431]. 2.3, 2.4, 3, 3

AUTHORS

Uriel Feige [About the author]
Microsoft Research
One Microsoft Way
Redmond, WA 98052-6399
and
Weizmann Institute of Science
Rehovot 76100, Israel
urifeige@microsoft.com
uriel.feige@weizmann.ac.il
http://www.wisdom.weizmann.ac.il/~feige

Eran Ofek [About the author]
Department of Computer Science
and Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel
eran.ofek@weizmann.ac.il
http://www.wisdom.weizmann.ac.il/~erano

ABOUT THE AUTHORS

URIEL FEIGE is a member of the theory group at Microsoft Research, currently on leave from the Weizmann Institute. His main research interests involve studying the borderline between P and NP as it manifests itself in approximation algorithms, heuristics, and exact algorithms that are not necessarily polynomial time. Other activities he enjoys include playing the piano, dancing with his wife, and helping his kids with their homework.

ERAN OFEK obtained his Ph. D. in Computer Science from the Weizmann Institute of Science in 2006. His Ph. D. and M. S. advisor was Uriel Feige. His research interests include optimization algorithms, random structures, and average case complexity. On his spare time he likes to play soccer, volleyball, or spend time with his two sons.